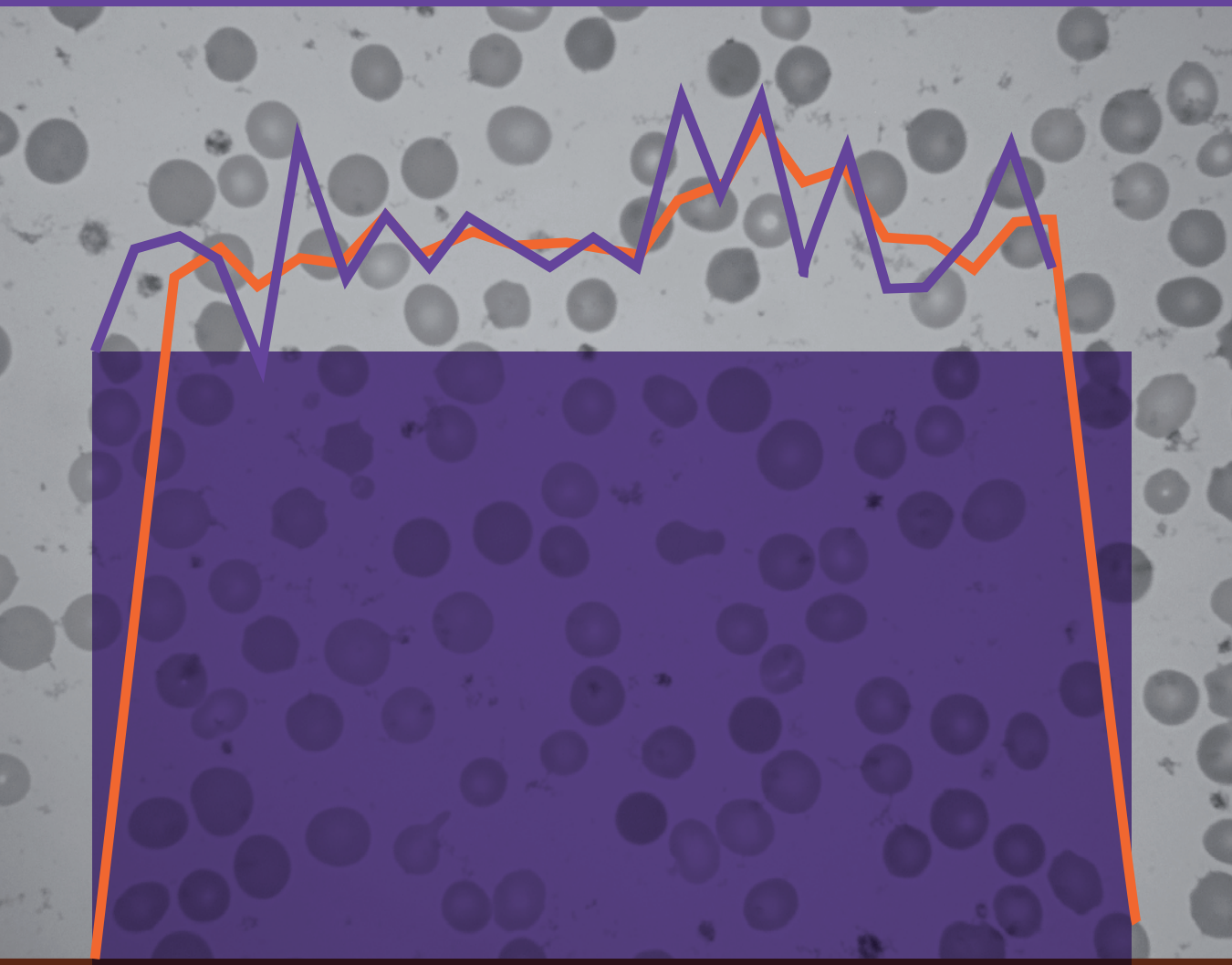


Lineární a adaptivní zpracování dat: řešené úlohy v MATLABu

Roman Vyškovský
Daniel Schwarz

Předmluva

Sbírka příkladů „Lineární a adaptivní zpracování dat: řešené úlohy v MATLABu“ vznikla v rámci řešení projektu MUNI/FR/1529/2016 „Fond rozvoje MU“ jako studijní opora k předmětu Lineární a adaptivní zpracování dat (Bi0440).

Množství dat, která reprezentují procesy, projevy a činnosti živých systémů, narůstá spolu s rapidním vývojem digitálních technologií, jež nám tato data umožňují pořizovat, přenášet a ukládat. Zvyšuje se tak i význam metod z oblasti digitálního zpracování a analýzy signálů, jejichž cílem je zvýrazňování signálu v šumu nebo transformace naměřených dat tak, aby mohly být objeveny jejich zdánlivě skryté vlastnosti. Náplní sbírky jsou příklady, které ukazují použití zmíněných metod na reálných nebo modelovaných datových souborech z oboru biologie, medicíny a environmentální oblasti. Vyřešením sbírky student lépe pochopí probíranou látku a získá schopnosti převádět často nesnadno uchopitelnou teorii do praxe. Praktická analýza dat se dnes již provádí výhradně pomocí výpočetní techniky s nástroji, jako jsou například MATLAB, R nebo Python.

Sbírka obsahuje 20 příkladů rozdělených do pěti kapitol kopírujících již zavedené dělení předmětu Lineární a adaptivní zpracování dat: 1) systémy, signály a časové řady, 2) lineární filtrace, 3) metody zvýraznění užitečné složky a potlačení šumu, 4) modely časových řad a 5) adaptivní zpracování a predikce časových řad. Ke každému z témat je vypracováno několik příkladů ve formě rozřešených skriptů a funkcí napsaných v Matlabu, které jsou volně ke stažení na oborovém portálu matematické biologie [1] a dále na výukovém portálu LF MU [2]. Každý příklad obsahuje dílčí úkoly, které má student za úkol doprogramovat podle pokynů vepsaných v komentářích. Správná řešení úkolů a objasnění prováděných výpočtů jsou uvedena v tomto dokumentu.

Sbírka příkladů slouží zejména pro samostudium a procvičování, ale také ke zlepšení kontaktní výuky v praktických cvičeních předmětu Lineární a adaptivní zpracování dat.

[1] http://portal.matematickabiologie.cz
[2] http://portal.med.muni.cz

Obsah

Pokyny k vypracování příkladů

Na jednom z níže uvedených odkazů si stáhněte složku s příklady:

Matematická biologie [1]

Portál Lékařské fakulty Masarykovy univerzity [2]

Postupně vypracujte příklady, které jsou v kořenové složce s názvem „priklad_“ (za podtržítkem následuje číslo kapitoly z učebnice Lineární a adaptivní zpracování dat (https://www.iba.muni.cz/res/file/ucebnice/schwarz-linearni-a-adaptivni-zpracovani-dat.pdf) a písmeno, které označuje pořadí příkladu. Každý soubor s příkladem obsahuje rozřešený kód s úkoly na doplnění částí kódu. Správné výsledky jsou uvedeny níže v tomto dokumentu. Složka s příklady dále obsahuje podsložky s daty a funkcemi. Součástí některých příkladů je doplnění kódu ve funkci.

[1] http://portal.matematickabiologie.cz/index.php?pg=analyza-a-modelovani-dynamickych-biologickych-dat--linearni-a-adaptivni-zpracovani-dat-resene-ulohy-v-matlabu

[2] http://portal.med.muni.cz/clanek-672-linearni-a-adaptivni-zpracovani-dat-resene-ulohy-v-matlabu.html

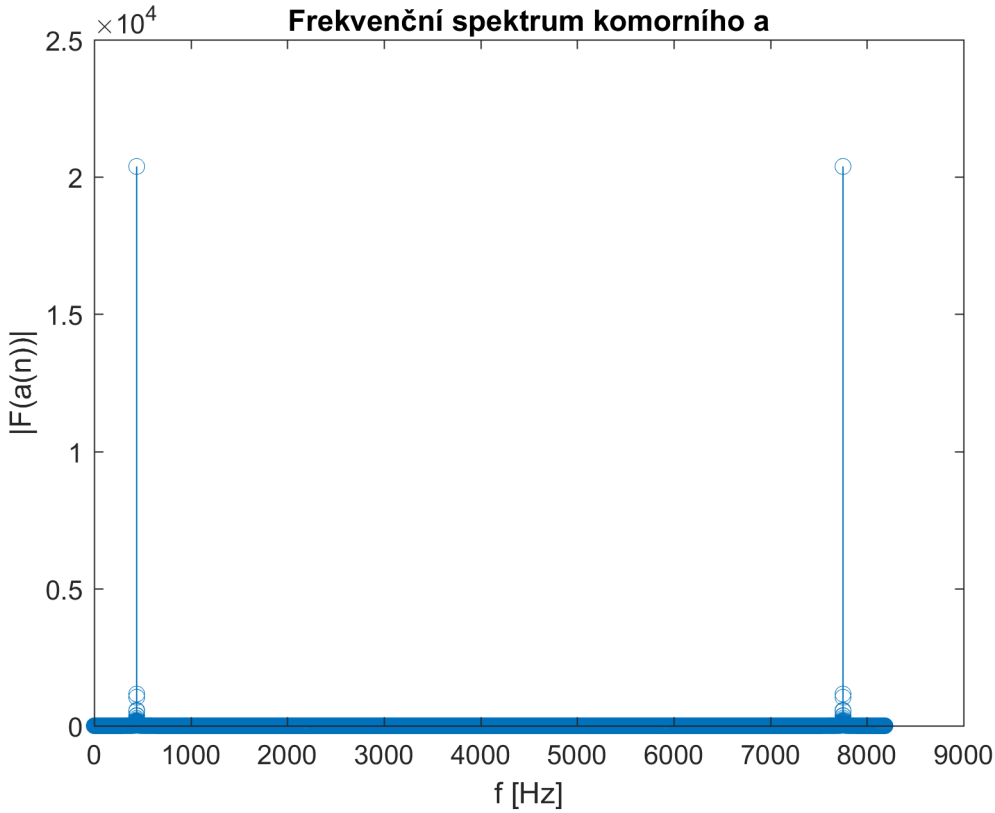
Kapitola 1: Systémy, signály a časové řady

Příklad 1A: Časové řady a jejich spektrum

Cílem příkladu je vytvořit harmonickou časovou řadu odpovídající tónu ‚komorní a', dále časovou řadu, která odpovídá součtu 3 tónů (C, E, G) a lze ji tak interpretovat jako akord C. Následně tyto řady převést do frekvenční domény a pozorovat, jak vypadá jejich spektrum. Nakonec pozorujte a interpretujte spektrum na reálném měření průměrné měsíční rychlosti větru v Brně Tuřanech mezi lety 1961–2016.

Úkol 1: Sestrojení tónu ‚komorní a'

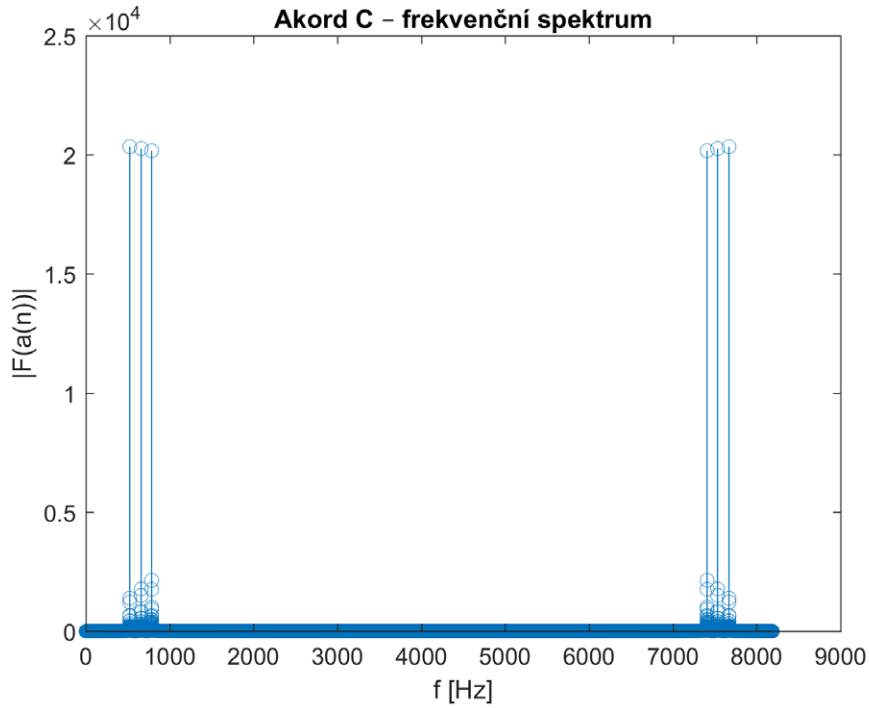
```
% Sestrojte periodicky signal odpovidajici tonu ‚komorni a' o frekvenci
% 440 Hz, vzorkovany frekvenci 8192Hz o delce 5s.

% Definujte vzorkovaci frekvenci:
FS = 8192;

% Vypocitejte vzorkovaci periodu:
Ts = 1/FS;

% Zvolte delku tonu 5s:
t = 0:Ts:5;

% Urcete frekvenci tonu:
A_FREQUENCY = 440;

% Pomoci parametru definovanych vyse a funkce sinus vytvorte ton komorni a:
a = sin(2*pi*A_FREQUENCY*t);
sound(a, FS)
% Pozn.: Funkce sound implicitne pocita s vzorkovaci frekvenci 8192Hz,
% pokud je ton vzorkovan touto frekvenci, lze parametr FS vynechat.

% Zobrazte frekvencni spektrum tonu a:
spectrum = fft(a);
lenSpe = length(spectrum);
x = 0:FS/lenSpe:(FS-FS/lenSpe);
stem(x, abs(spectrum)); % Obrazek 1A1
title(‚Frekvenční spektrum komorního a'); xlabel(‚f [Hz]'); ylabel(‚|F(a(n))|');
```



Obrázek 1A1: Frekvenční spektrum komorního a

Úkol 2: Sestrojte akord C, který se skládá z tónů C, E a G o frekvencích 523Hz, 659Hz a 784Hz

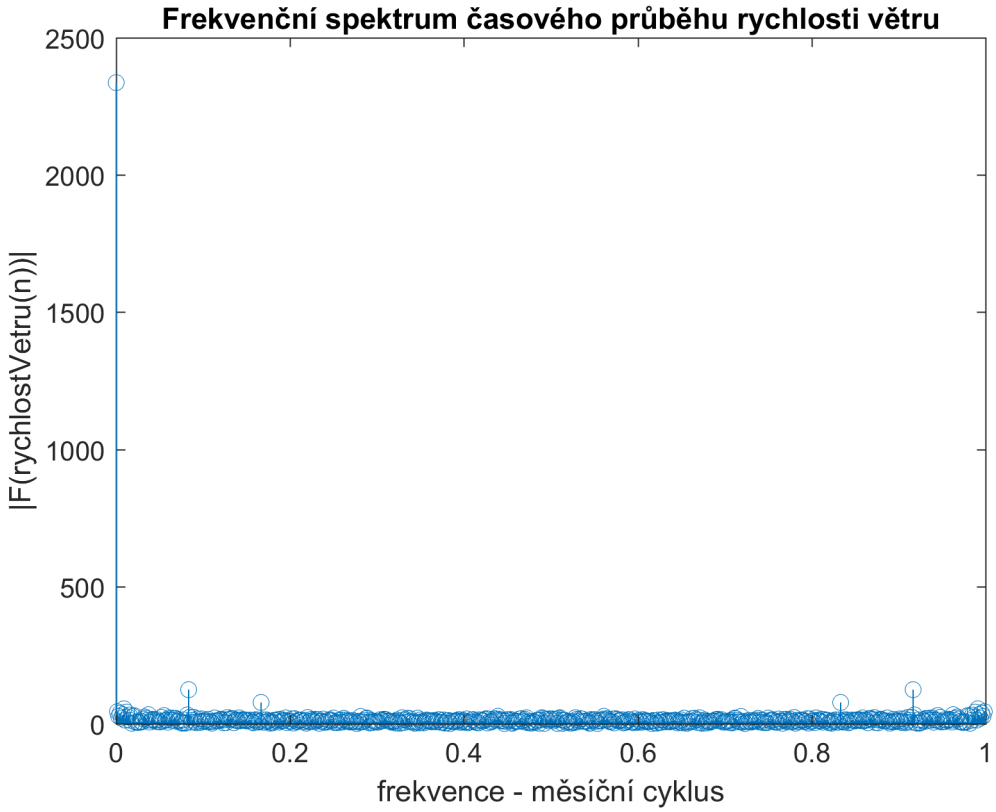
```
% Definujte frekvence tonu c, e a g:
C_FREQUENCY = 523;
E_FREQUENCY = 659;
G_FREQUENCY = 784;

% Sestrojte akord C:
chordC = sin(2*pi*C_FREQUENCY*t) + sin(2*pi*E_FREQUENCY*t)
+ sin(2*pi*G_FREQUENCY*t);
sound(chordC)

% Prohlednete si jeho frekvencni spektrum:
spectrum = fft(chordC);
lenSpe = length(spectrum);
x = 0:FS/lenSpe:(FS-FS/lenSpe);
stem(x, abs(spectrum)); % Obrazek 1A2
title('Akord C – frekvenční spektrum'); xlabel('f [Hz]'); ylabel('|F(a(n))|');
```



Obrázek 1A2: Frekvenční spektrum akordu C

Úkol 3: Spektrum průměrné měsíční rychlosti větru v Brně-Tuřanech mezi lety 1961–2016

```
% Odhadnete jake frekvence obsahuje casova rada ‚\data\prumerna_mesicni_rychlost_vetru_turany.csv‘:
% Zdroj dat: http://portal.chmi.cz/historicka-data/pocasi/denni-data#
% (Prumerna denni rychlost vetru byla prevedena na prumernou mesicni
% rychlost vetru.)
windSpeed = csvread(char(strcat(pwd, ‚\data\prumerna_mesicni_rychlost_vetru_turany.csv‘)));
stem(windSpeed), title(‚Průměrná měsíční rychlost větru‘), xlabel(‚rok‘), ylabel(‚v [m/s]‘)
set(gca,‘xtick‘,1:48:length(windSpeed),‘xticklabel‘,num2cell(1961:4:2016))

% Doplnte vzorkovaci frekveci:
FS = 1; % 1 vzorek za mesic

% Aplikujte rychlou fourierovu transformaci:
spectrum = fft(windSpeed);

% Vykreslete frekvencni spektrum a interpretujte:
lenSpe = length(spectrum);
x = 0:FS/lenSpe:(FS-FS/lenSpe);
stem(x, abs(spectrum)); % Obrazek 1A3
title(‚Frekvenční spektrum časového průběhu rychlosti větru‘); xlabel(‚frekvence - měsíční cyklus‘); ylabel(‚|F(rychlostVetru(n))|‘);
% Casova rada obsahuje periodicke deje s pulrocni (f = 0.1667) a rocni periodou (f = 0.08333).
```



Obrázek 1A3: Frekvenční spektrum rychlosti větru

Příklad 1B: Základní popis systému

Určete diferenční rovnici, impulzní charakteristiku a přenosovou funkci klouzavého aritmetického průměru o 3 vzorcích. Vykreslete jeho frekvenční charakteristiku a vyšetřete jeho stabilitu (ve frekvenční i časové oblasti) a nakonec určete jeho vlastnosti (kauzalita, časová invariantnost, linearita).

Úkol 1: Určete diferenční rovnici:

```
% y(k) = (1/3)*x(k)+(1/3)*x(k-1)+(1/3)*x(k-2);
```

Úkol 2: Napište impulzní charakteristiku:

```
h = [1/3 1/3 1/3];
```

Úkol 3: Určete přenosovou funkci:

```
% H(z) = (1/3)+(1/3)*z^(-1)+(1/3)*z^(-2);
```

Úkol 4: Nakreslete frekvenční charakteristiku:

```
[H,w] = freqz(h,1,'whole'); % whole - na cele jednotkove kruznici
plot(w,abs(H)); % Obrazek 1B1
title('Frekvenční charakteristika')
xlabel('\omega [rad/s]')
ylabel('|G(\omega)|')
```

Úkol 5: Komentujte stabilitu systému:

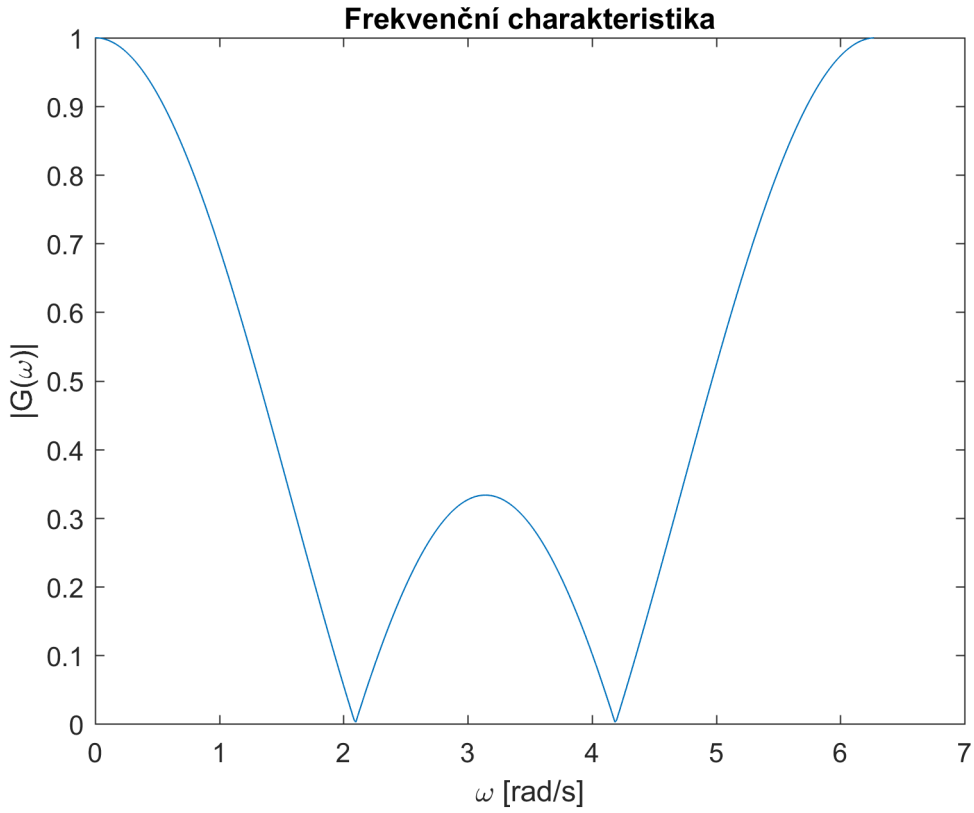
```
% a) V časové doméně:
% Podle konecneho souctu vzorku impulzni charakteristiky je zrejme, ze je
% system stabilni.
if (sum(abs(h))<Inf)
   disp(‚Stabilní‘)
else
   disp(‚Nestabilní‘)
end

% b) Ve frekvenční doméně:
zplane(h,1) % Vsechny poly prenosove funkce jsou umisteny v jednotkove kruznici (Obrazek
1B2)
xlabel(‚Re‘), ylabel(‚Im‘)
```

Úkol 6: Je systém…

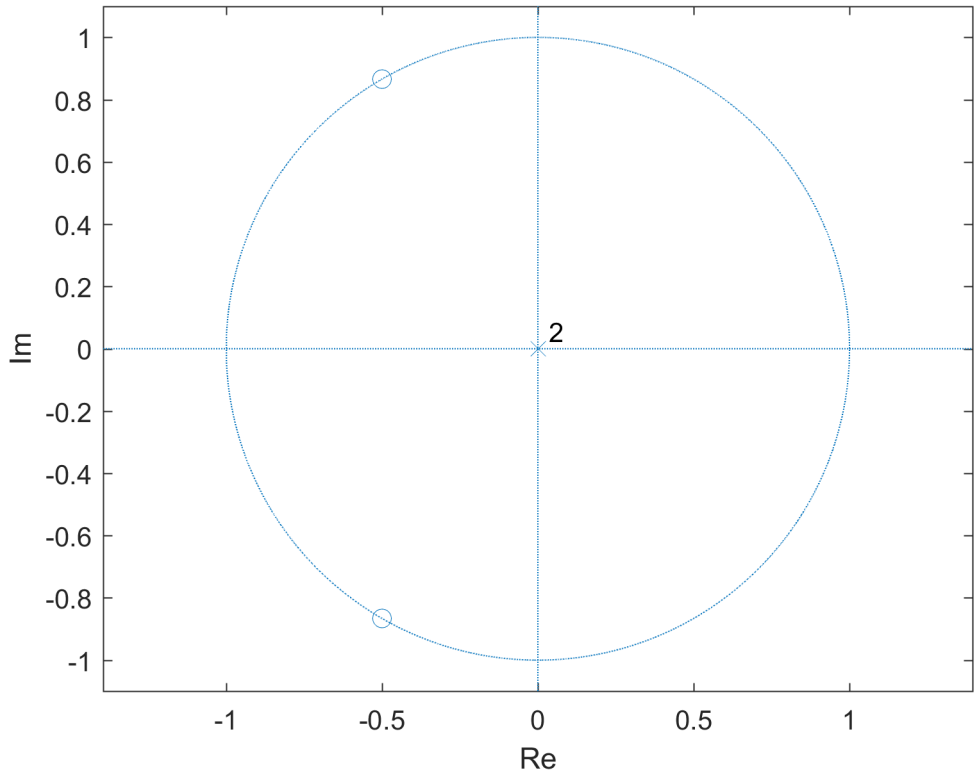
…kauzální? ANO – chování systému závisí pouze na současné hodnotě a minulých hodnotách

…časově invariantní? ANO – chování systému se v čase nemění

…lineární? ANO – systém popisují lineární diferenční rovnice, lze uplatnit princip superpozice



Obrázek 1B1: Frekvenční charakteristika



Obrázek 1B2: Jednotková kružnice

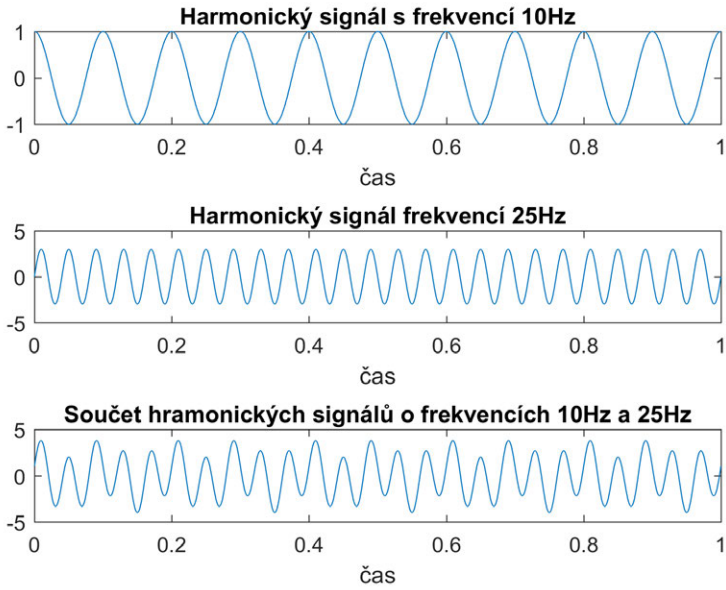
Příklad 1C: Diskretizace signálu a aliasing

Uvažujme úlohu, ve které chceme navzorkovat signál, a získat tak časové řady pro další zpracovaní. Cílem úkolu bude identifikovat aliasing a určit správnou vzorkovací frekvenci.

Úkol 1: Vytvoření časových řad

```
% Vytvorte casove rady s frekvenci 10Hz a 25Hz a jejich soucet:

% Casova rada 1: frekvence = 10Hz, amplituda = 1, delka trvani = 1s, pocet
% vzorku 1000
FREQUENCY_1 = 10;
t = linspace(0,1,1000);
y1a = cos(2*pi*FREQUENCY_1*t);
subplot(3,1,1) % Obrazek 1C1
plot(t,y1a)
title('Harmonický signál s frekvencí 10Hz'), xlabel('čas')

% Casova rada 2: frekvence = 25Hz, amplituda = 3, delka trvani = 1s, pocet
% vzorku 1000
FREQUENCY_2 = 25;
t = linspace(0,1,1000);
y2a = 3*sin(2*pi*FREQUENCY_2*t);
```

```
subplot(3,1,2)
plot(t,y2a)
title('Harmonický signál frekvencí 25Hz'), xlabel('čas')

% Casova rada 3: soucet obou casovych rad
ya = y1a + y2a;
subplot(3,1,3)
plot(t,ya)
title('Součet hramonických signálů o frekvencích 10Hz a 25Hz'), xlabel('čas')
```

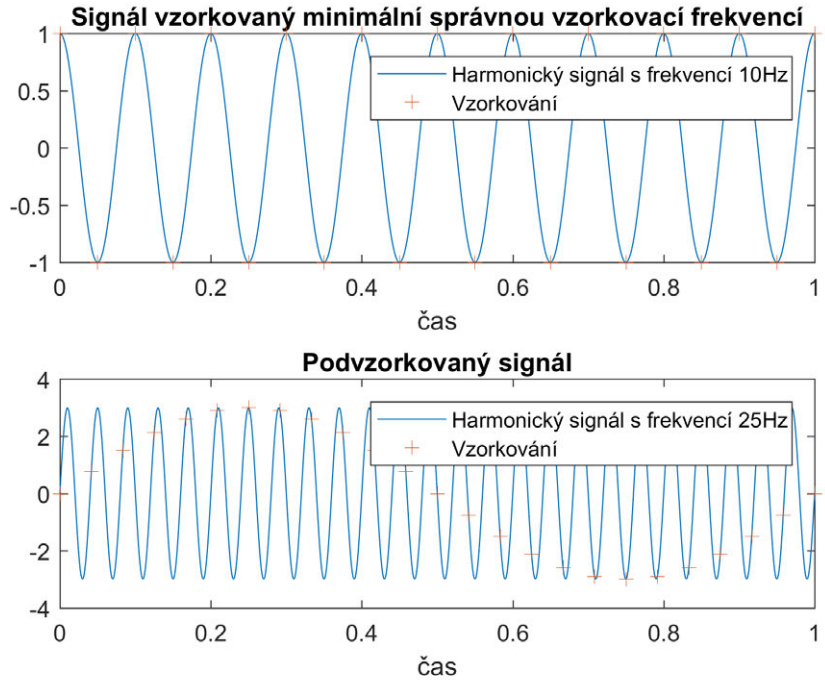


Obrázek 1C1: Harmonické signály (10Hz, 25Hz) a jejich součet

Úkol 2: Identifikace aliasingu

```
% Urcete zda jsou nasledujici pokusy o navzorkovani uspesne. Je signal
% podvzorkovany? Pozorujte aliasing.

% Vzorkovani 1
figure, subplot(2,1,1) % Obrazek 1C2
plot(t,y1a), hold on
sampling = linspace(0,1,21);
y1b = cos(2*pi*FREQUENCY_1*sampling);
plot(sampling,y1b,'+')
title('Signál vzorkovaný minimální správnou vzorkovací frekvencí'), xlabel('čas')
legend({'Harmonický signál s frekvencí 10Hz','Vzorkování'})
hold off
% Casova rada 1 je vzorkovana vice jak 2 vzorky za periodu, neni
% podvzorkovana.

% Vzorkovani 2
subplot(2,1,2)
plot(t,y2a), hold on
```

```
sampling = linspace(0,1,25);
y2b = 3*sin(2*pi*FREQUENCY_2*sampling);
plot(sampling,y2b,'+')
title('Podvzorkovaný signál'), xlabel('čas')
legend({'Harmonický signál s frekvencí 25Hz','Vzorkování'})
% Casova rada 2 je vzorkovana 1 vzorkem za periodu, je podvzorkovana.
```

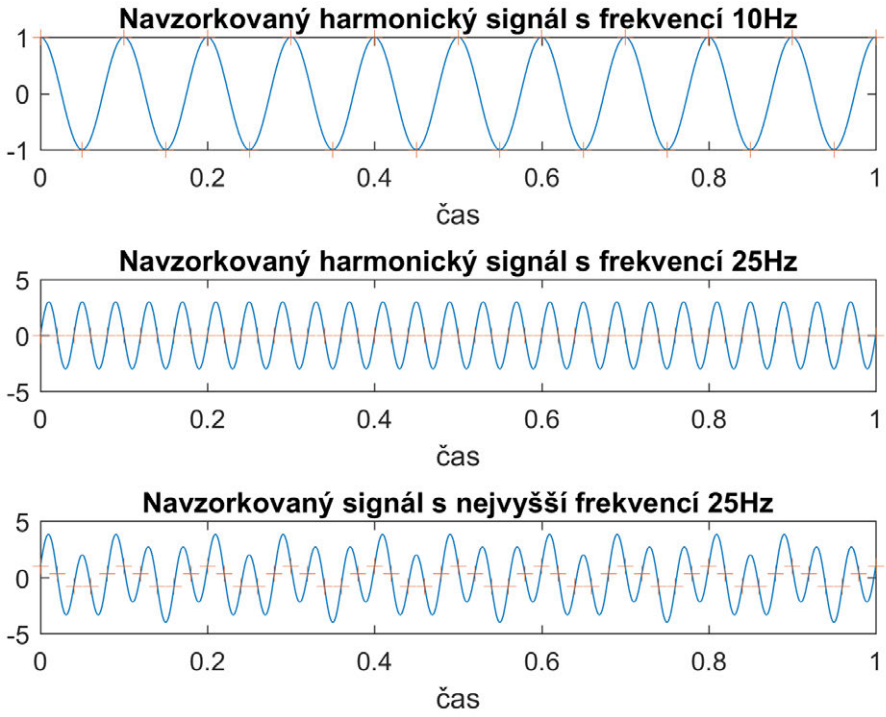


Obrázek 1C2: Správně vzorkovaný signál a podvzorkovaný signál

Úkol 3: Vzorkování

```
% Nasledujici signaly vzorkujte minimalni spravnou vzorkovaci frekvenci:

% Aby byl signal spravne navzorkovany, musi byt vzorkovaci frekvence vyssi
% nez dvojnasobek frekvence harmonicke slozky o nejvyssi frekvenci, ktera
% se nachazi ve vzokovanem signalu.

% Casova rada 1 s frekvenci 10Hz
figure, subplot(3,1,1) % Obrazek 1C3
plot(t,y1a), hold on
sampling = linspace(0,1,21);
% Alespon 21 vzorku zajisti splneni vzorkovaciho teoremu
y1b = cos(2*pi*FREQUENCY_1*sampling);
plot(sampling,y1b,'+'), xlabel('čas')
title('Navzorkovaný harmonický signál s frekvencí 10Hz')

% Casova rada 2 s frekvenci 25Hz
subplot(3,1,2)
plot(t,y2a), hold on
sampling = linspace(0,1,51);
```

```
% Alespon 51 vzorku zajisti splneni vzorkovaciho teoremu
y2b = 3*sin(2*pi*FREQUENCY_2*sampling);
plot(sampling,y2b,'+'), xlabel('čas')
title('Navzorkovaný harmonický signál s frekvencí 25Hz')

% Soucet obou casovych rad
subplot(3,1,3)
plot(t,ya), hold on
sampling = linspace(0,1,51);
% Alespon 51 vzorku zajisti splneni vzorkovaciho teoremu, nebot harmonicka
% slozka s frekvenci 25Hz je nejvyssi.
yb = cos(2*pi*FREQUENCY_1*sampling) + 3*sin(2*pi*FREQUENCY_2*sampling);
plot(sampling,yb,'+'), xlabel('čas')
title('Navzorkovaný signál s nejvyšší frekvencí 25Hz')
```



Obrázek 1C3: Vzorkování signálu

Příklad 1D: Výpočet konvoluce

Vytvořte funkci pro výpočet konvoluce a aplikujte ji na vyhlazení časové řady měření diastolického krevního tlaku.

Úkol 1: Načtení dat

```
% Nactete data z data\diastolicky_tlak.mat
% Diastolicky krevni tlak [mm Hg] mereny po hodine behem dne:
load(char(strcat(pwd, ',\data\diastolicky_tlak.mat')));
% Popiste o jaka jde data:
% jednorozmerna, nezavisla promenna je cas
```

Úkol 2: Impulzní charakteristika

```
% Do vektoru ‚h' ulozte impulzni charakteristiku klouzaveho aritmetickeho
% prumeru o delce 3 vzorky:
h = [1/3 1/3 1/3];

% Popiste, o jaky jde system:
% kauzalni, casove invariantni, linearni
```

Úkol 3: Doplnění funkce ‚convolution'

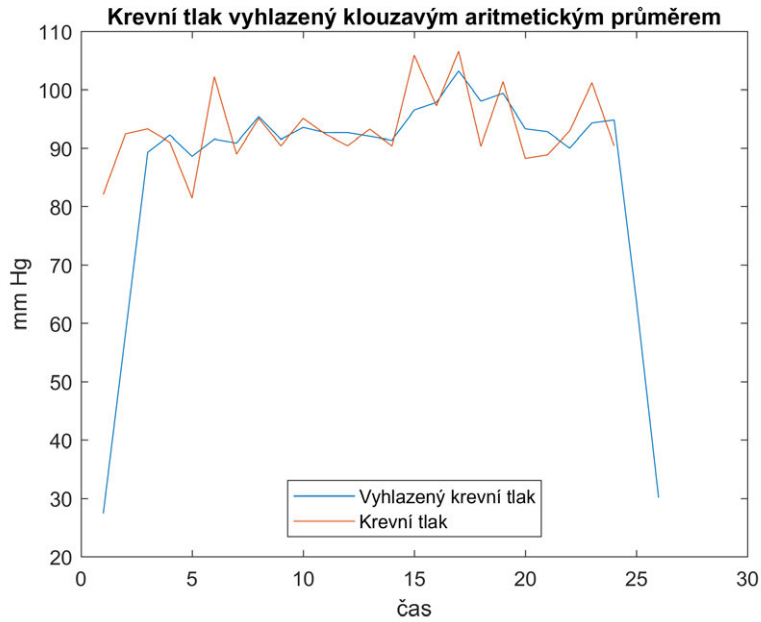
```
% Doplnte vzorec pro konvoluci do funkce funkce\convolution.m
% Zajistete, aby byla funkce pristupna tomuto skriptu:
cd(‚funkce');
```

Úkol 4: Aplikace konvoluce

```
% Vypocitejte odezvu systemu popsaneho impulzni charakteristikou z ukolu 2
% na vstup signalu z ukolu 1:
result = convolution(diastolicPressure, h);
```

Úkol 5: Vykreslení výsledku

```
% Vykreslete vysledek, porovnejte s puvodni casovou radou a diskutujte
% rozdilnou delku obou casovych rad.
figure, plot(result) % Obrazek 1D1
hold on
plot(diastolicPressure)
xlabel(‚čas')
ylabel(‚mm Hg')
title(‚Krevní tlak vyhlazený klouzavým aritmetický průměrem')
legend({‚Vyhlazený krevní tlak', ‚Krevní tlak'}, ‚location', ‚south')

% Delka vysledku konvoluce je dana vzorcem: n+m-1, kde n - pocet vzorku
% casove rady, m - pocet vzorku konvolucniho jadra. Vysledek je dale mozne
% orezat a ponechat jen validni vzorky (napr. u funkce conv argument ‚valid').
```



Obrázek 1D1: Výsledek vyhlazení diastolického krevního tlaku

Funkce: convolution

```
function result = convolution(timeSeries, kernel)
% convolution - funkce pro pocitani konvoluce diskretniho signalu a
%           konvolucniho jadra
%
% result = convolution(timeSeries, kernel)

% Deklarujte:
% n - pocet vzorku casove rady
% m - pocet vzorku konvolucniho jadra
n = length(timeSeries);
m = length(kernel);
stepsNum = n + m - 1;
result = zeros(1,stepsNum);

  for i = 1:stepsNum

    helper = 0;
    for j = max(1,i+1-m):1:min(i,n)

      % Doplnte vzorec pro konvoluci:
      helper = helper + timeSeries(j) .* kernel(i-j+1);

    end

    result(i) = helper;
  end
end
```

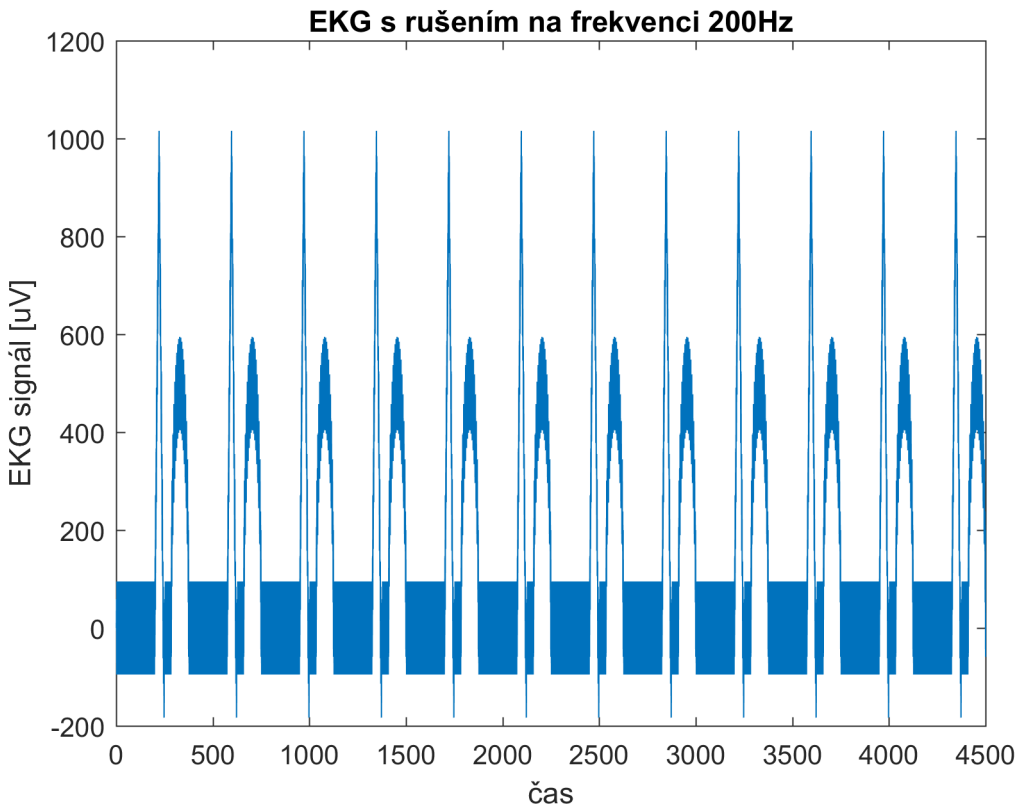
Kapitola 2: Lineární filtrace

Příklad 2A: Návrh filtru metodou okénka

Navrhněte FIR filtr pro odstranění rušení z dat EKG. Pro návrh filtru použijte metodu okénka.

Úkol 1: Načtení dat

```
% Nactete data \data\ekg.mat
% Data byla vymodelovana pomoci nastroje:
% https://www.physionet.org/physiotools/matlab/ECGwaveGen/
% Frekvence vzorkovani = 500Hz
% Ruseni na frekvenci 200Hz.
load(char(strcat(pwd, ,\data\ekg.mat‘)));
FS = 500;
plot(ecg) % Obrazek 2A1
title(‚EKG s rušením na frekvenci 200Hz‘), ylabel(‚EKG signál [uV]‘), xlabel(‚čas‘)
```

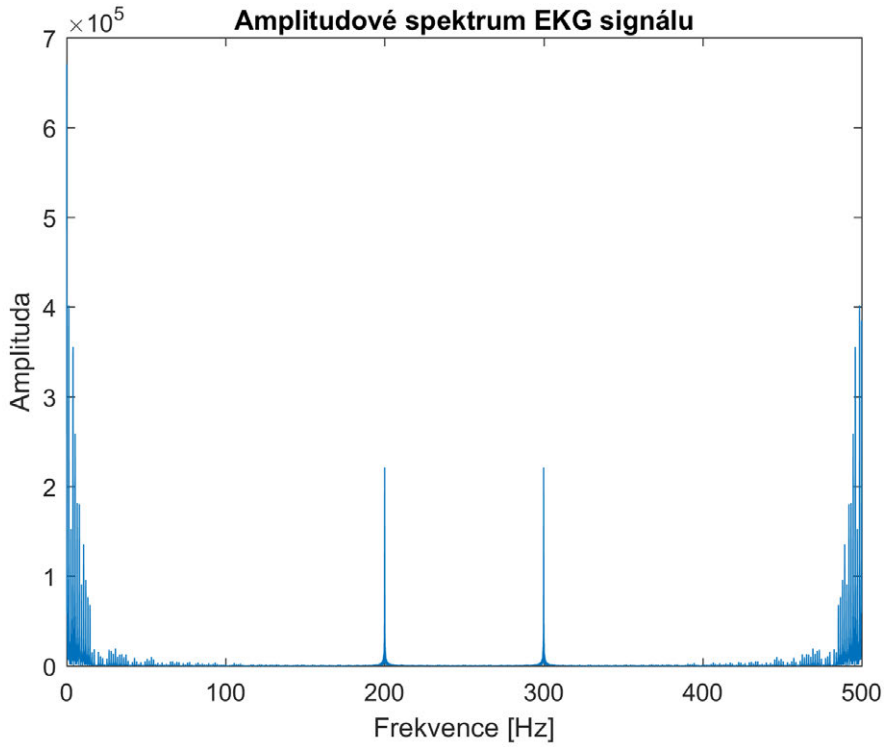


Obrázek 2A1: EKG s rušením

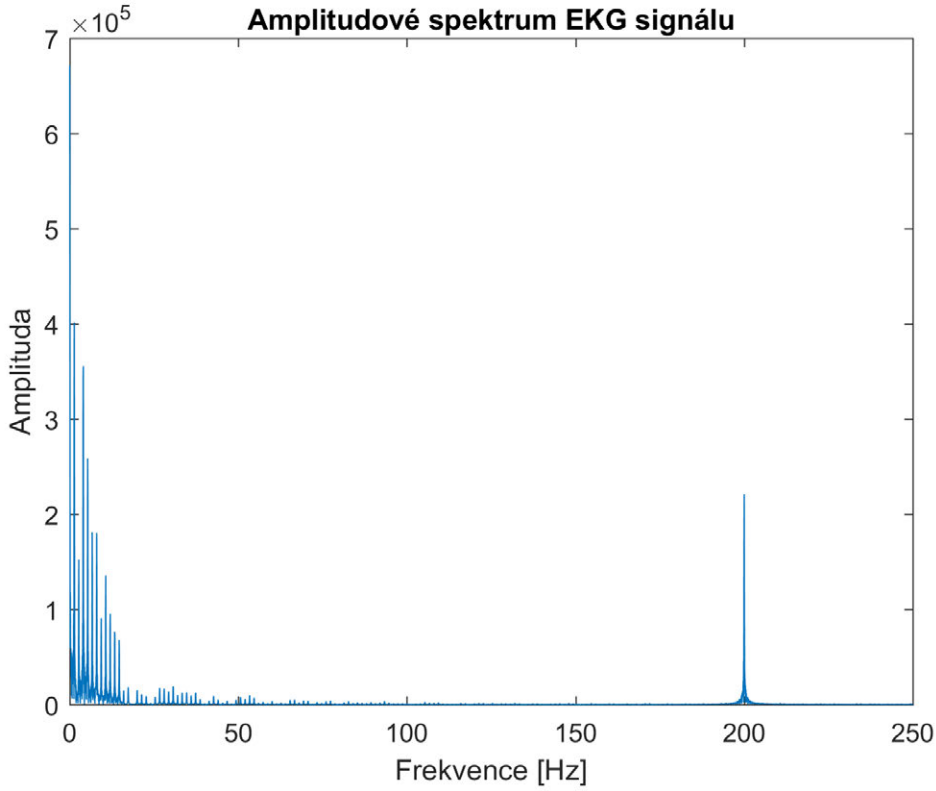
Úkol 2: Spektrum signálu

```
% Nejprve se podivejte na spektrum signalu:
samplesNum = length(ecg);
samplesFftNum = 2.^nextpow2(samplesNum);
```

```
% Pozn.: 2.^nextpow2(x) najde prvni mocninu 2 na nejake cislo takovou, aby
% byl vysledek vyssi nez x. FFT se dela pro takto definovany pocet vzorku,
% a jeji vypocet je rychlejsi.

% Pro vypocet frekvenčního spektra pouzijte diskretni Fourierovu
% transformaci (DFT) signalu, a to pomoci algoritmu rychle DFT:
spectrum = fft(ecg, samplesFftNum);
% Pozn. pokud je ‚samplesFftNum' vyssi nez pocet vzorku signalu, pak je
% pro vypocet DFT signal doplnen nulami

% Vykreslete amplitudovou cast frekvencniho spektra (vysledkem DFT je
% komplexni posloupnost).
x = 0:FS/samplesFftNum:(FS-FS/samplesFftNum);
figure, plot(x, abs(spectrum)) % Obrazek 2A2
title(‚Amplitudové spektrum EKG signálu')
ylabel(‚Amplituda'), xlabel(‚Frekvence [Hz]')

% Z grafu lze pozorovat symetrii spektra, pro veskerou informaci staci
% vykreslit polovina grafu:
spectrum = spectrum(1:samplesFftNum/2);
x = 0:FS/2/(samplesFftNum/2):(FS/2-FS/2/(samplesFftNum/2));
figure, plot(x, abs(spectrum)) % Obrazek 2A3
title(‚Amplitudové spektrum EKG signálu')
ylabel(‚Amplituda'), xlabel(‚Frekvence [Hz]')
% Pozn: V grafu je videt uzkopasmove ruseni na frekvenci 200Hz, proto
% v dalsim kroku vytvorime filtr, ktery odstrani frekvence vyssi nez 190Hz.
```



Obrázek 2A2: Amplitudové spektrum EKG



Obrázek 2A3: Amplitudové spektrum EKG (postačující graf)

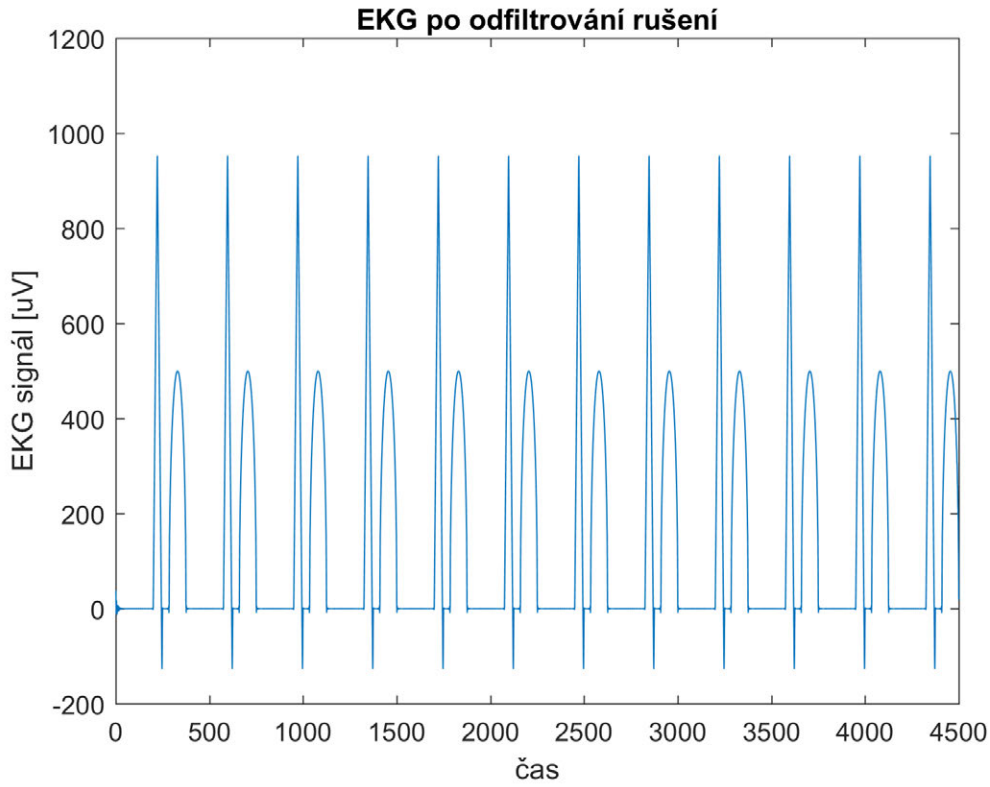
Úkol 2: Návrh filtru

```
% Pouzijte funkci ‚fir1‘, ktera vytvari FIR filtr metodou okenka, nastavte
% rad filtru a cut-off frekvenci. Funkce vraci koeficienty impulzni
% charakteristiky.

% Do promenne ‚tresholdFreq‘ ulozte frekvenci 190Hz normalizovanou na
% Nyqistovu frekvenci.
tresholdFreq = 190/(FS/2);

% Nastavte promenou ‚rad‘ definujici rad filtru:
rad = 100; % Napriklad filtr radu 100

% Vytvorte filtr:
h = fir1(rad, tresholdFreq);
```

Úkol 3: Aplikace filtru

```
% Pro ocisteni signalu pouzijte konvoluci s impulzni charakteristikou
% vytvoreneho FIR filtru.
ecgPure = conv(ecg,h,‘same‘);
% Parametr ‚same‘ urcuje vraceni pouze centralni casti konvoluce tak, aby

% delka vystupni casove rady odpovidala delce vstupni casove rady.
```

```
% Zobrazte ocisteny signal:
figure, plot(ecgPure) % Obrazek 2A4
title(‚EKG po odfiltrování rušení‘), ylabel(‚EKG signál [uV]‘), xlabel(‚čas‘), ylim([-200 1200])
```

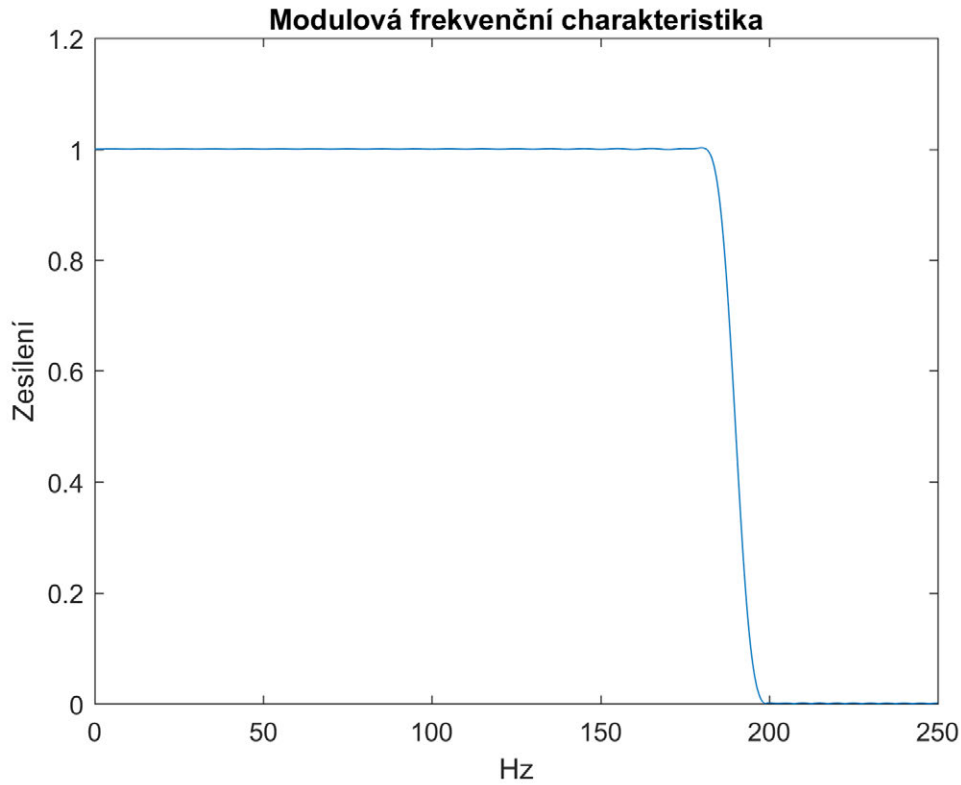


Obrázek 2A4: EKG očištěné od rušení

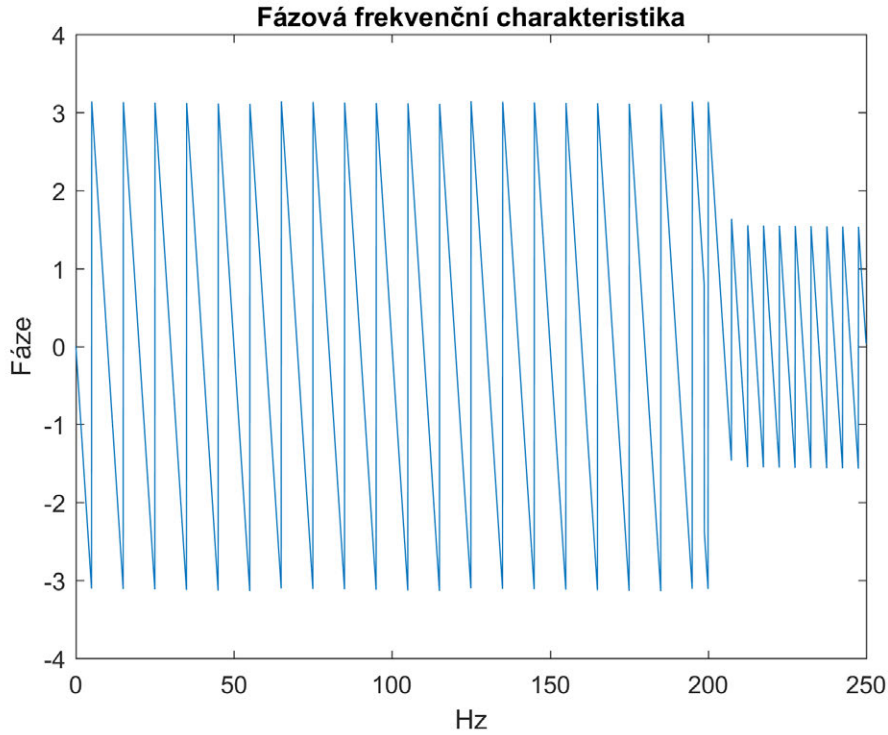
Úkol 4: Vlastnosti filtru

```
% Vypocitejte frekvencni charakteristiku filtru (amplitudovou i fazovou):
fCh = fft(h, samplesFftNum);
fCh = fCh(1:samplesFftNum/2);

% Doplnte kod pro vykresleni amplitudove a fazove frekvencni
% charakteristiky:
figure, plot(x, abs(fCh)) % Obrazek 2A5
title(‚Modulová frekvenční charakteristika‘), ylabel(‚Zesílení‘), xlabel(‚Hz‘)
% Filtr tlumi signal na frekvenci 200Hz

figure, plot(x, angle(fCh)) % Obrazek 2A6
title(‚Fázová frekvenční charakteristika‘), ylabel(‚Fáze‘), xlabel(‚Hz‘)

% A pro uplnost overte jeho stabilitu. Jde o FIR filtr, tedy musi byt
% stabilni. Póly jsou uvnitř jednotkové kružnice.
figure, zplane(fft(h),1) % Obrazek 2A7
xlabel(‚Re‘), ylabel(‚Im‘)
```

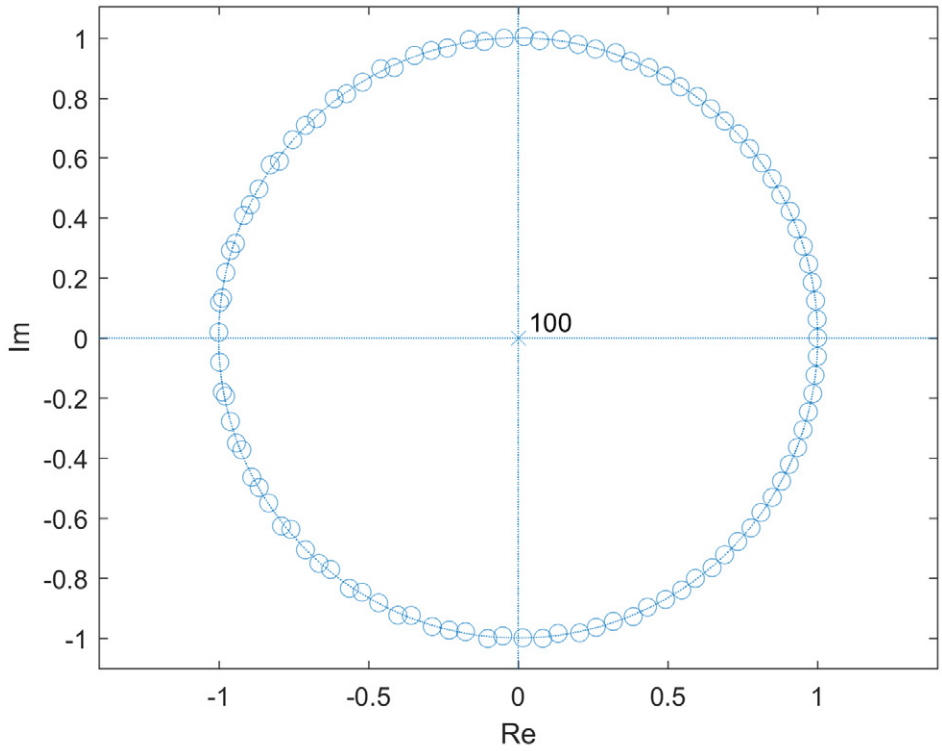
```
% Pozn. Filtrace signalu lze provest i ve frekvencni domene (naznacuje
% nasledujici graf).
s = abs(spectrum);
s = s/max(s(:));
plot(x,s), hold on, plot(x,abs(fCh)); % Obrazek 2A8
legend({‚Amplitudové spektrum EKG signálu (normalizované)‘,‘Modulová frekvenční
charakteristika FIR filtru‘});
xlabel(‚Hz‘);

freqDomainFiltering = fCh.*spectrum; % Nasobeni ve frekvencni domene odpovida konvoluci
v casove domene.
figure, plot(x, abs(freqDomainFiltering)); % Obrazek 2A9
title(‚Amplitudové spektrum očištěného EKG signálu‘), xlabel(‚Hz‘), ylabel(‚Amplituda‘)
```



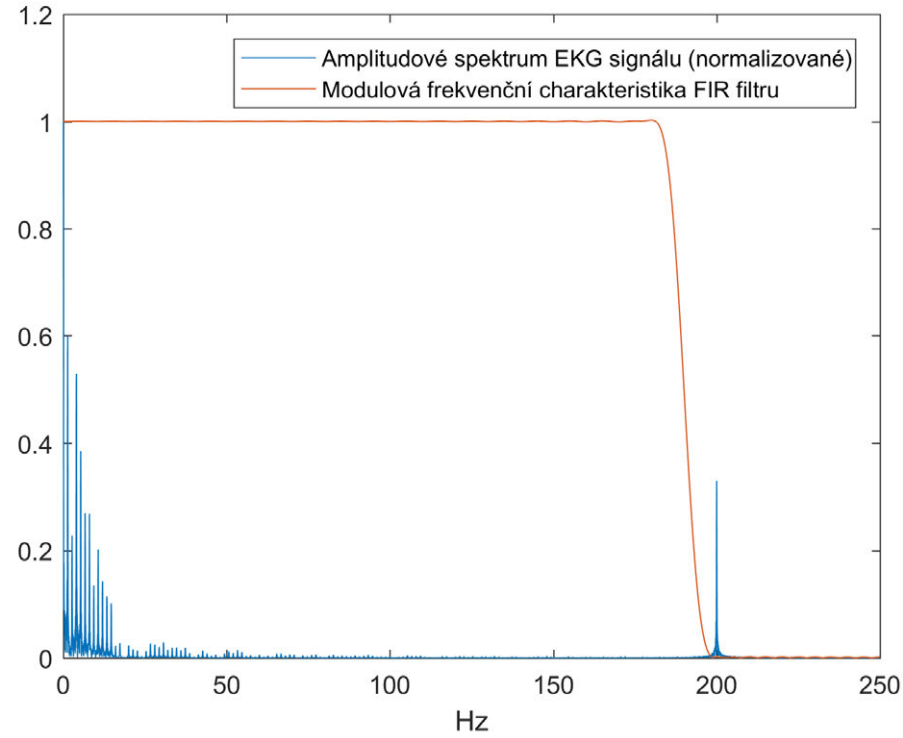
Obrázek 2A5: Modulová frekvenční charakteristika filtru



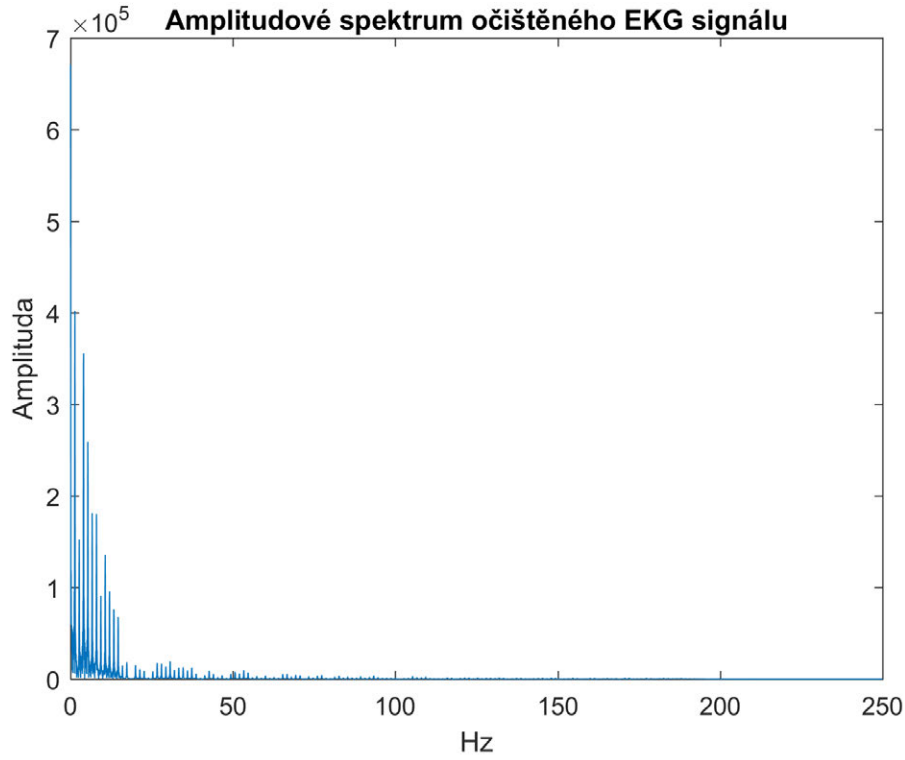
Obrázek 2A6: Fázová frekvenční charakteristika filtru



Obrázek 2A7: Rozložení nul a pólů



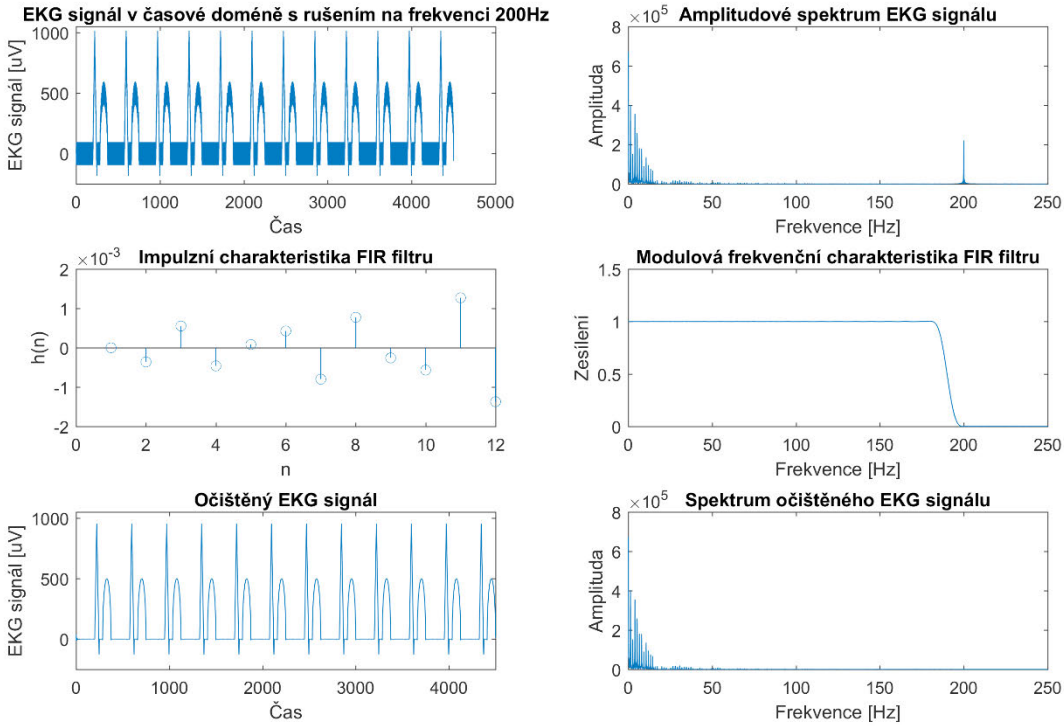
Obrázek 2A8: Filtrace ve frekvenční doméně



Obrázek 2A9: Amplitudové spektrum EKG po filtraci

Úkol 5: Tvorba maticového grafu

```
% Na zaver udelejte maticovy graf o dimenzi 3×2, kde v levem sloupci bude
% casova domena a v pravem frekvencni domena. V 1. radku puvodni signal,
% ve druhem popis filtru (impulzni, frekvencni charakteristiky) a ve 3. radku
% filtrovany signal.
figure, subplot(3,2,1);
plot(ecg); % Obrazek 2A10
title('EKG signál v časové doméně s rušením na frekvenci 200Hz');
ylabel('EKG signál [uV]');
xlabel('Čas');
ylim([-250, 1050]);

subplot(3,2,2);
plot(x, abs(spectrum));
title('Amplitudové spektrum EKG signálu');
ylabel('Amplituda');
xlabel('Frekvence [Hz]');

subplot(3,2,3);
stem(h);
title('Impulzní charakteristika FIR filtru');
ylabel('h(n)');
xlabel('n');
xlim([0, 12]);

subplot(3,2,4);
plot(x, abs(fCh));
title('Modulová frekvenční charakteristika FIR filtru');
ylabel('Zesílení');
xlabel('Frekvence [Hz]');

subplot(3,2,5);
plot(ecgPure);
title('Očištěný EKG signál');
ylabel('EKG signál [uV]');
xlabel('Čas');
ylim([-250, 1050]);
xlim([0, 4500]);

subplot(3,2,6);
plot(x, abs(freqDomainFiltering));
title('Spektrum očištěného EKG signálu');
ylabel('Amplituda');
xlabel('Frekvence [Hz]');
```



Obrázek 2A10: Amplitudové spektrum EKG po filtraci

Příklad 2B: Návrh filtru metodou vzorkování frekvenční charakteristiky

Navrhněte FIR filtr pro odstranění rušení z dat EKG. Pro návrh filtru použijte metodu vzorkování frekvenční charakteristiky.

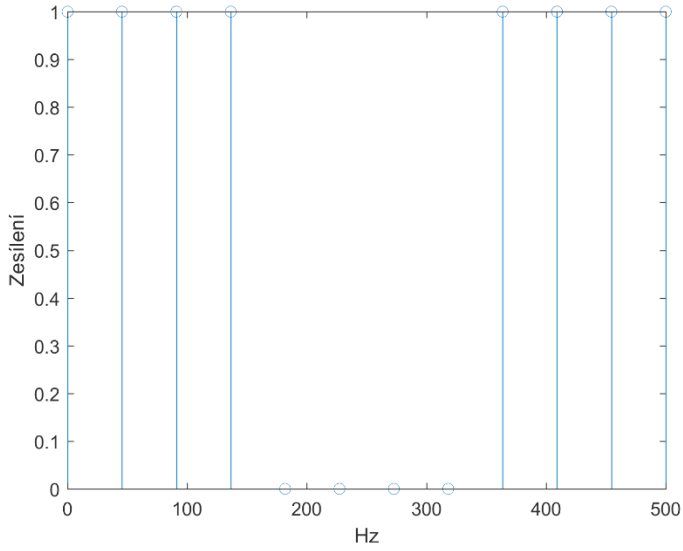
Úkol 1: Načtení dat

```
% Nactete data \data\ekg.mat
% Data byla vymodelovana pomoci nastroje:
% https://www.physionet.org/physiotools/matlab/ECGwaveGen/
% Frekvence vzorkovani = 500Hz
% Ruseni na frekvenci 200Hz.
load(char(strcat(pwd, ',\data\ekg.mat')));
FS = 500;
```

Úkol 2: Návrh frekvenční charakteristiky filtru

```
% Navrhnete frekvencni charakteristiku filtru typu dolni propust pro odstraneni ruseni na
% frekvenci 200Hz. Pro srovnani zvolte stejny rad filtru jako u predchoziho prikladu
% metody.
% Napr.: Pomocny graf:
stem(0:(FS/11):FS, ones(1,12)), xlabel('Hz'), ylabel('Zesílení')
```

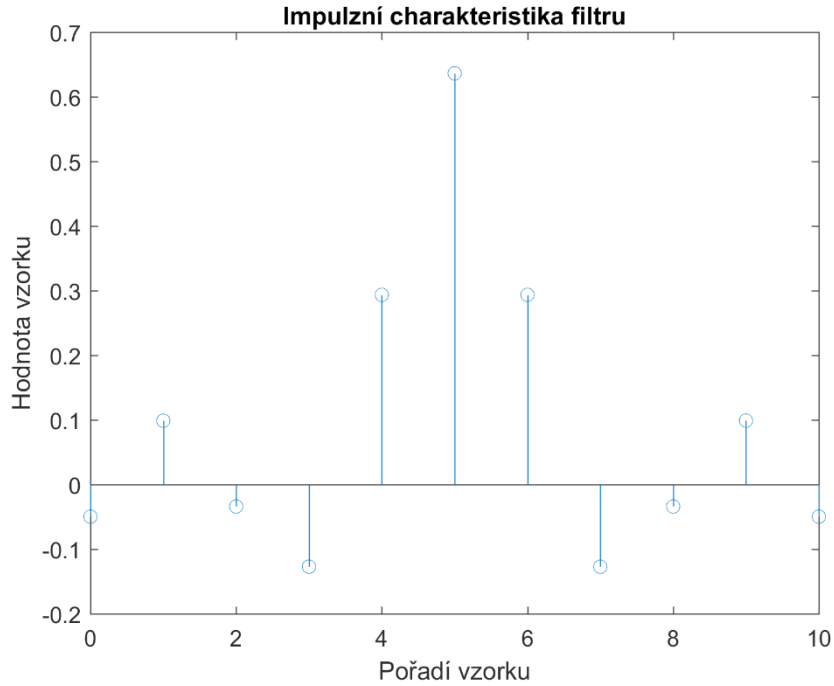
```
% Vzorkovani frekvencni charakteristiky volime tak, aby hodnota fr.char.
% na frekvenci 200Hz nabyvala hodnoty 0.
freqChar = ones(1,11);
freqChar(5:8) = 0;
figure, stem(0:(500/11):500, [freqChar 1]), xlabel(‚Hz‘), ylabel(‚Zesílení‘) % Obrazek 2B1
% Ziskali jsme navzorkovanou frekvencni charakteristiku filtru, ktery
% odstrani ruseni na frekvenci 200Hz.
```



Obrázek 2B1: Vzorkování frekvenční charakteristiky

Úkol 3: Vytvoření filtru

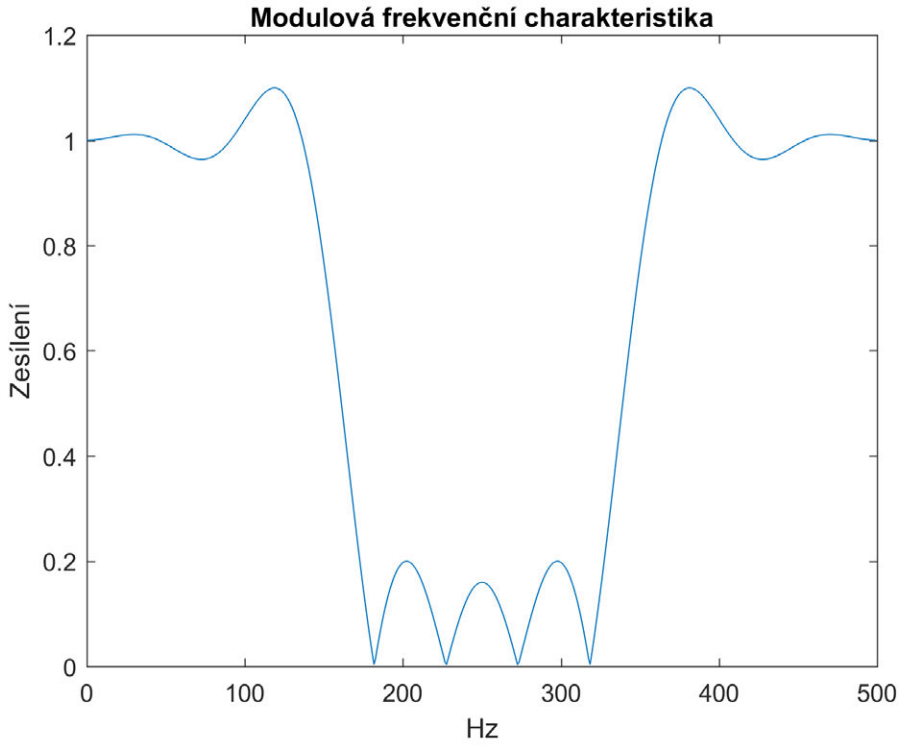
```
% Provedte zpetnou Fourierovu transformaci frekv. char. a ziskejte impulzni
% charakteristiku filtru:
h = ifft(freqChar);

% „Abychom ziskali kauzalni filtr a symetrickou impulzni charakteristiku, je
% potreba zpozdit jeji vzorky o (N-1)/2.“
% Citace: Ing. Jiří Kozumplík, CSc., Ing. Radim Kolář, Prof. Ing. Jiří Jan, CSc., Číslicové
zpracování signálů v prostředí Matlab. Skriptum, 2001.
h = [h(ceil(length(h)/2)+1:end) h(1:ceil(length(h)/2))];
figure, stem(0:length(h)-1,h) % Obrazek 2B2
title(‚Impulzní charakteristika filtru‘)
xlabel(‚Pořadí vzorku‘), ylabel(‚Hodnota vzorku‘)
```



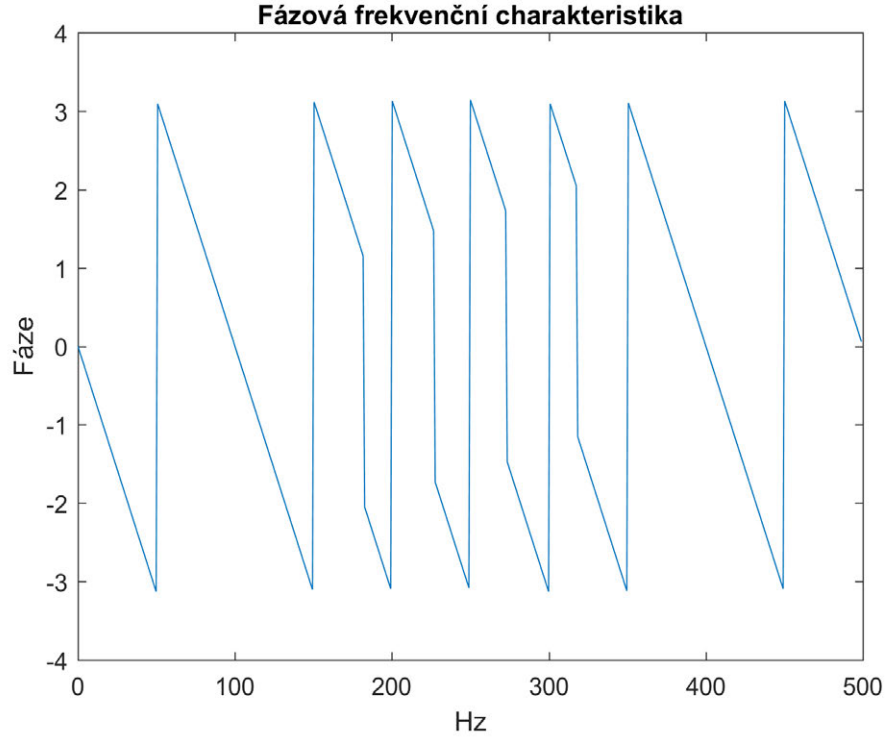
Obrázek 2B2: Impulzní charakteristika filtru

Úkol 4: Frekvenční charakteristika a stabilita filtru

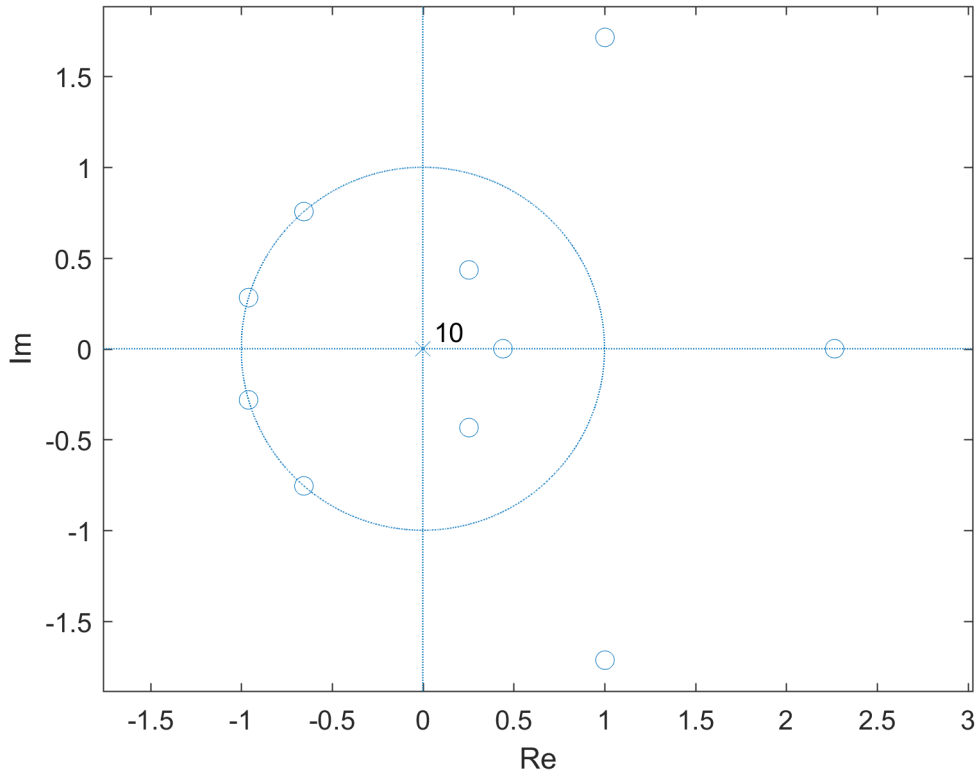
```
% Vykreslete spojitou modulovou a fazovou frekvencni charakteristiku filtru.
[G,w] = freqz(h,1,'whole');
x = w/2/pi*500;
plot(x,abs(G)) % Obrazek 2B3
title('Modulová frekvenční charakteristika'), xlabel('Hz'), ylabel('Zesílení')
% Pozn.: Vidite, ze dochazi k tlumeni na frekvenci 200Hz. Neni to nicmene uplne
% utlumeni na dane frekvenci. Pro zlepseni kvality je napr. mozne zvysit
% rad filtru.

% Protoze impulzni charakteristika je symetricka, ma FIR filtr linearni
% fazovou charakteristiku.
figure, plot(x,angle(G)) % Obrazek 2B4
title('Fázová frekvenční charakteristika'), xlabel('Hz'), ylabel('Fáze')

% Pro overeni stability zkotrolujte rozlozeni polu:
figure, zplane(h,1); % Poly jsou v jednotkove kruznici, filtr je stabilni. Obrazek 2B5
xlabel('Re'), ylabel('Im')
```



Obrázek 2B3: Modulová frekvenční charakteristika filtru



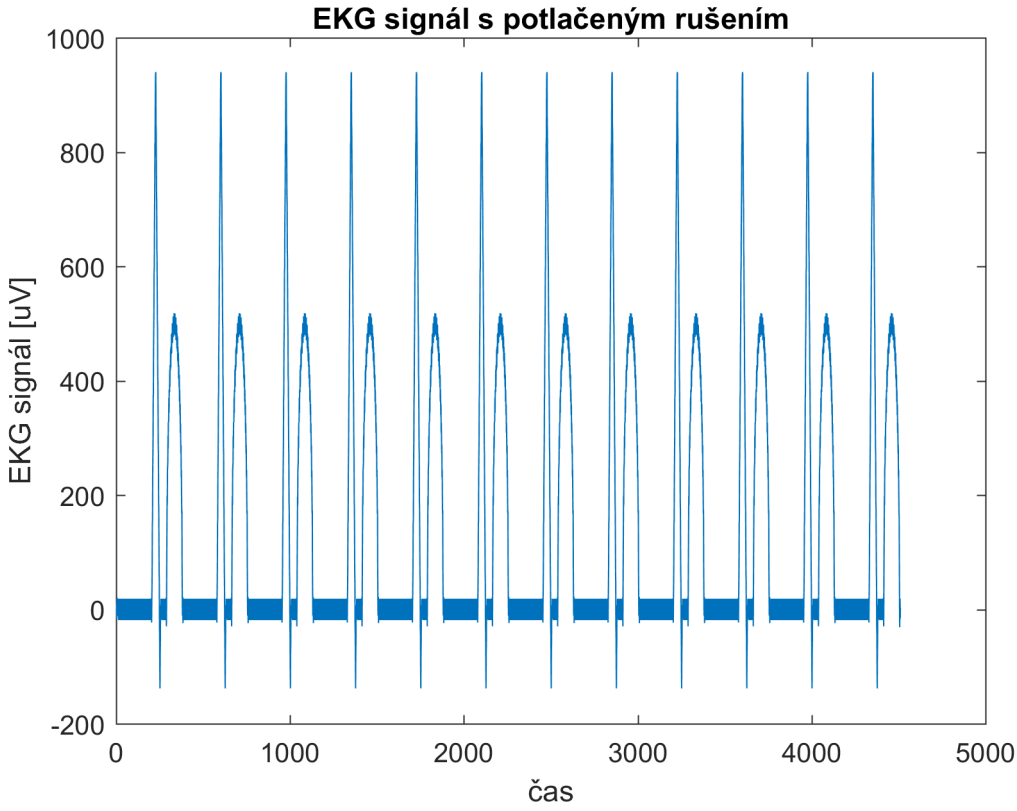
Obrázek 2B4: Fázová frekvenční charakteristika filtru



Obrázek 2B5: Rozložení nul a pólů

Úkol 5: Aplikování filtru

```
% Nakonec ocistete signal EKG pomoci filtru od sumu:
ecgPure = conv(ecg,h);
plot(ecgPure) % Obrazek 2B6
title(‚EKG signál s potlačeným rušením‘), xlabel(‚čas‘), ylabel(‚EKG signál [uV]‘)
```



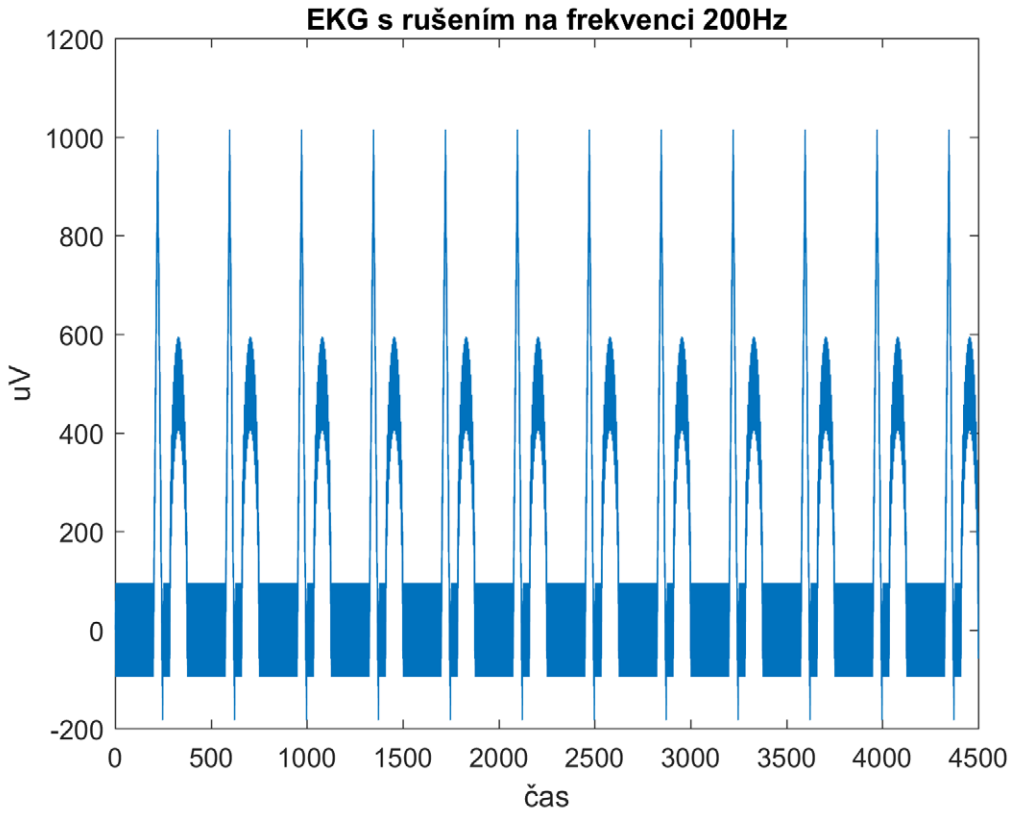
Obrázek 2B6: EKG s potlačeným rušením

Příklad 2C: Návrh IIR filtru

Navrhněte IIR filtr pro odstranění šumu z EKG signálu.

Úkol 1: Načtení dat

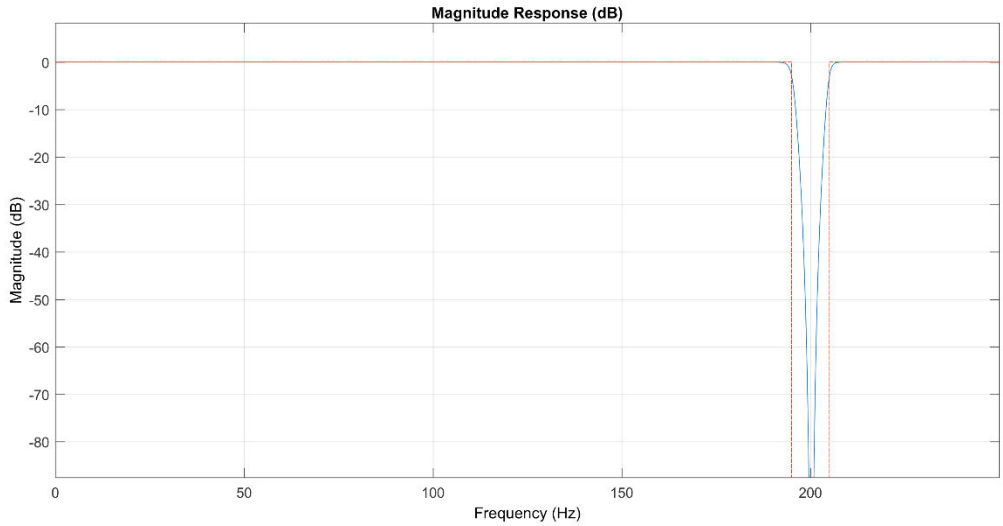
```
% Nactete data \data\ekg.mat:
% Data byla vymodelovana pomoci nastroje:
% https://www.physionet.org/physiotools/matlab/ECGwaveGen/
% Frekvence vzorkovani = 500Hz
% Zasumeni na frekvenci 200Hz.
load(char(strcat(pwd, ,\data\ekg.mat‘)));
plot(ecg) % Obrazek 2C1
xlabel(‚čas‘), ylabel(‚uV‘), title(‚EKG s rušením na frekvenci 200Hz‘)
```



Obrázek 2C1: EKG s rušením

Úkol 2: Návrh filtru typu pásmová zádrž

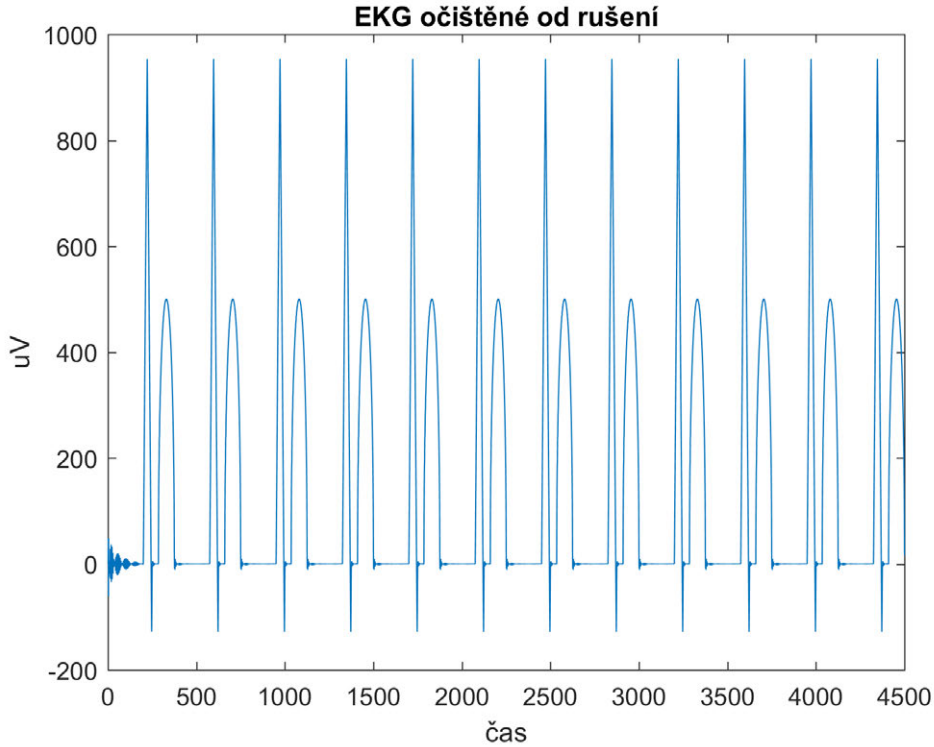
```
% Vzorkovaci frekvence
FS = 500;
RAD = 10;
CUTOFF_LOWER = 195;
CUTOFF_HIGHER = 205;

% Pomoci fdesign.bandstop navrhnete IIR filtr typu pasmova zadrz:
bandstopIIR = fdesign.bandstop(‚N,F3dB1,F3dB2‘, RAD, CUTOFF_LOWER, CUTOFF_HIGHER,
FS);
% Dokumentace: https://www.mathworks.com/help/dsp/ref/fdesign.bandstop.html

% Definujte Butterworthuv filtr:
butterBandstopIIR = design(bandstopIIR,‘butter‘);

% Pomoci jedineho prikazu sledujte graficky charakteristiky filtru:
fvtool(butterBandstopIIR) % Obrazek 2C2
% Pomoci ‚fvtool‘ lze pozorovat napriklad rozlozeni nul a polu, impulzni
% nebo frekvencni charakteristiky filtru
```



Obrázek 2C2: FVTOOL

Úkol 3: Aplikace filtru

```
% Aplikujte filtr na data
result = filter(butterBandstopIIR, ecg);
plot(result) % Obrazek 2C3
xlabel('čas'), ylabel('uV'), title('EKG očištěné od rušení')
```



Obrázek 2C3: EKG očištěné od rušení

Příklad 2D: Mediánový filtr pro odstranění šumu z obrazu

Navrhněte mediánový filtr a následně aplikujte na obrazová data.

Úkol 1: Načtení obrazu

```
% Nactete soubor \data\krvinky.jpg:
imgRgb = imread(strcat(pwd, ',\data\krvinky.jpg'));
imshow(imgRgb)
```

Úkol 2: Úprava obrazu

```
% Prevedte obraz na sedotonovy:
imgGrey = rgb2gray(imgRgb);
imshow(imgGrey)

% Zaruste obraz pomoci Gaussovskeho sumu:
imgGreyNoise = imnoise(imgGrey, 'gaussian');
figure, imshow(imgGreyNoise)
```

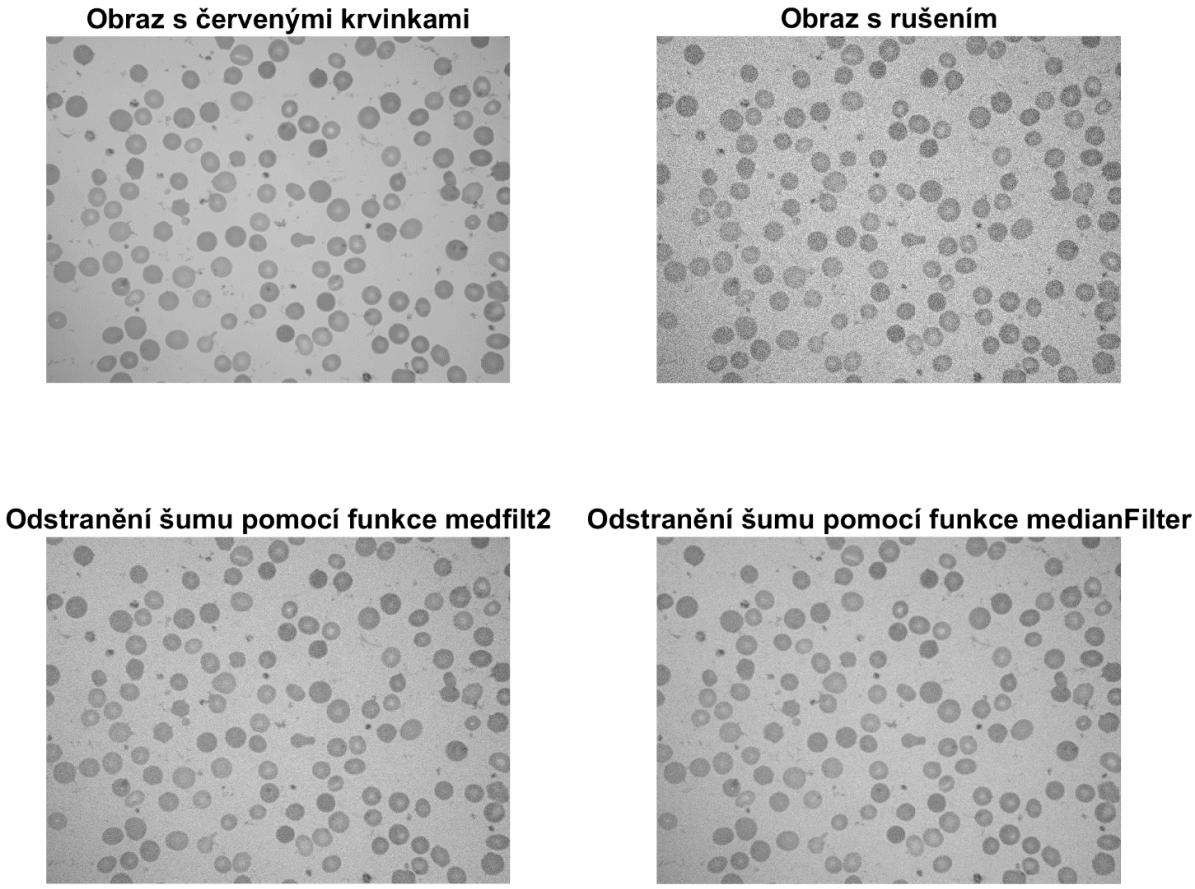
Úkol 3: Použití mediánového filtru

```
% Odstrante sum pomoci medianoveho filtru s dimenzi 3×3, ktery je
% implementovany v Matlabu ve funkci 'medfilt2':
filterDimension = [3 3];
imgGreyMedian = medfilt2(imgGreyNoise, filterDimension);

% Odstrante sum pomoci vlastniho medianoveho filtru s dimenzi 3×3, ktery vytvorite
% ve funkci 'medianFilter' ve slozce funkce.
cd('funkce')
imgGreyMedianCustom = medianFilter(imgGreyMedian, filterDimension);
```

Úkol 4: Srovnání obrazů v grafu

```
% Vykrelete do maticoveho grafu 4×4..

% ..puvodni obraz:
subplot(2,2,1) % Obrazek 2D1
imshow(imgGrey);
title('Obraz s červenými krvinkami');

% ..obraz zaruseny Gaussovskym sumem:
subplot(2,2,2)
imshow(imgGreyNoise)
title('Obraz s rušením');

% ..ocisteny obraz pomoci funkce 'medfilt2':
subplot(2,2,3)
imshow(imgGreyMedian)
title('Odstranění šumu pomocí funkce medfilt2');
```

```
% ..ocisteny obraz pomoci funkce ‚medianFilter‘:
subplot(2,2,4)
imshow(imgGreyMedianCustom);
title(‚Odstranění šumu pomocí funkce medianFilter‘);
```



Obrázek 2D1: Filtrace obrazu mediánovým filtrem

Funkce: medianFilter

```
function result = medianFilter(image, filterDimension)
% medianFilter - Funkce pocita vysledek medianoveho filtru, ktery je
%           limitovan jen na liche filtry. Okraje obrazu jsou
%           doplnene nulami.
%
% result = medianFilter(image, filterDimension)
% Dimenze filtru napr. [3 3]
```

```
% Testujte pocet vstupnich parametru:
if nargin < 1
   error('Vložte obrázek.')
elseif nargin < 2
   filterDimension = [3 3];
   disp('Dimenze filtru definována vektorem [3 3]');
end

% Testujte spravnou dimenzi filtru a obrazu:
if any(mod(filterDimension,2)==0)
  error('Dimenze filtru musí být liché');
elseif length(filterDimension)~=2
    error('Filtr musí být 2D.');
elseif length(size(image))~=2
    error('Obraz musí být 2D.');
end

% Ulozte dimenze filtru do promennych dimFil1 a dimFil2:
[m,n] = size(image);
dimFil1 = filterDimension(1);
dimFil2 = filterDimension(2);

% Doplnte okraje obrazku nulami tak, aby se dala vypocitat konvoluce pro
% cely obraz. Pro zjednoduseni predpokladejme, ze filtr bude lichy a bude
% mit velikost napriklad 3×3:
imgMargin = zeros(m+dimFil1-1, n+dimFil2-1); % pro sudy filtr by se pouzilo: zeros(m+dim1,
n+dim2);
o = size(imgMargin,1)/2-m/2;
p = size(imgMargin,2)/2-n/2;
imgMargin(o+1:o+m,p+1:p+n) = image;

result = zeros(m, n);
for i = 1:m
   for j = 1:n

      % Do promenne 'subImage' doplnte usek obrazu o velikosti filtru,
      % ktery odpovida iteraci i,j:
      subImage = imgMargin(i:i+dimFil1-1, j:j+dimFil2-1);

      % Z podobrazu vypocitejte median a ulozte ho na spravne misto v
      % matici 'result':
      result(i,j) = median(subImage(:));

   end
end

result = uint8(result); % Prevod obrazu do uint8, aby fungovalo vykresleni

end
```

Kapitola 3: Metody zvýraznění užitečné složky a potlačení šumu

Příklad 3A: +/- průměrování

Pomocí metody +/- průměrování odhadněte vlastnosti šumu v MRI datech obrazů mozků pacientů se schizofrenií. Jde o šum systematicky generovaný rušivými elementy v místnosti, kde došlo k akvizici obrazu, nebo jde o bílý šum?

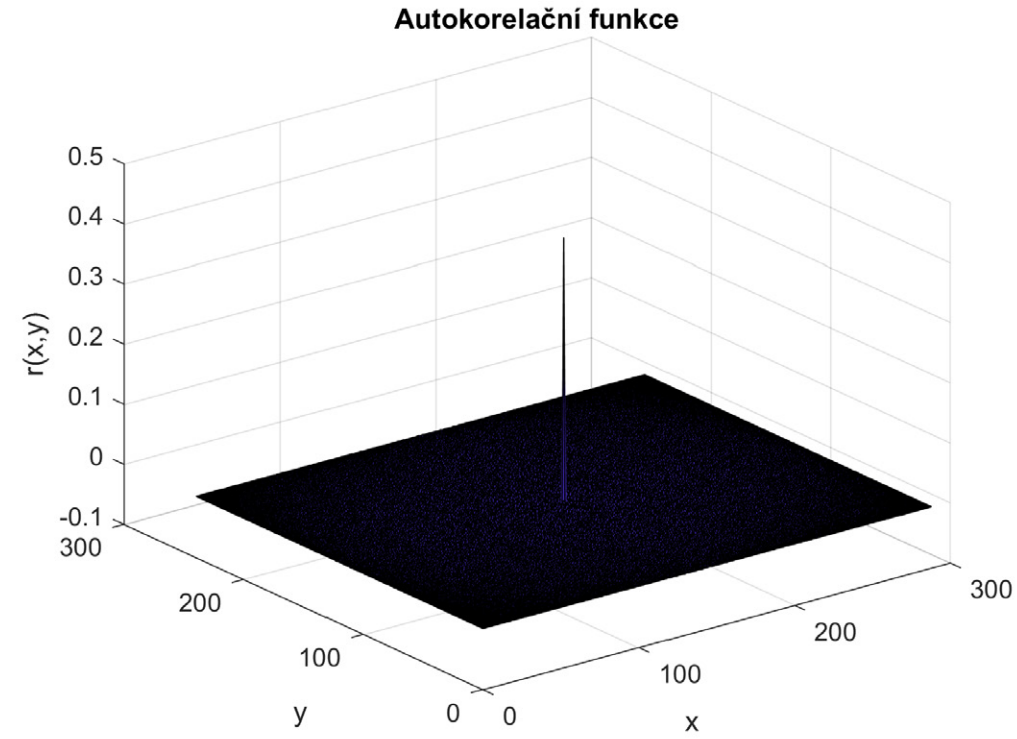
Úkol 1: Načtení dat

```
% Nacteni 20 obrazu mozku z cesty ‚\data\mozky_gm' s nazvem ‚gmA_*' a
% ulozeni do promenne ‚brainsA'.
i=1; % Nejprve se nacte obraz 1, aby se zjistily jeho rozmery a mohly se pouzit pro definovani
% velikosti matice ‚brainsA'.
load(char(strcat(pwd, ‚\data\mozky_gm\gmA_',num2str(i),'.mat')));
brainsA = zeros([size(gmA_1),20]);
brainsA(:,:,i) = eval(strcat(‚gmA_', num2str(i)));
clear(strcat(‚gmA_', num2str(i)))
for i = 2:20
   load(char(strcat(pwd, ‚\data\mozky_gm\gmA_',num2str(i),'.mat')));
   brainsA(:,:,i) = eval(strcat(‚gmA_', num2str(i)));
   clear(strcat(‚gmA_', num2str(i)))
end

% Nactete 20 obrazu mozku z cesty ‚\data\mozky_gm' s nazvem ‚gmB_*' a
% ulozte do promenne ‚brainsB'.
i=1;
load(char(strcat(pwd, ‚\data\mozky_gm\gmB_',num2str(i),'.mat')));
brainsB = zeros([size(gmB_1),20]);
brainsB(:,:,i) = eval(strcat(‚gmB_', num2str(i)));
clear(strcat(‚gmB_', num2str(i)))
for i = 2:20
   load(char(strcat(pwd, ‚\data\mozky_gm\gmB_',num2str(i),'.mat')));
   brainsB(:,:,i) = eval(strcat(‚gmB_', num2str(i)));
   clear(strcat(‚gmB_', num2str(i)))
end
```

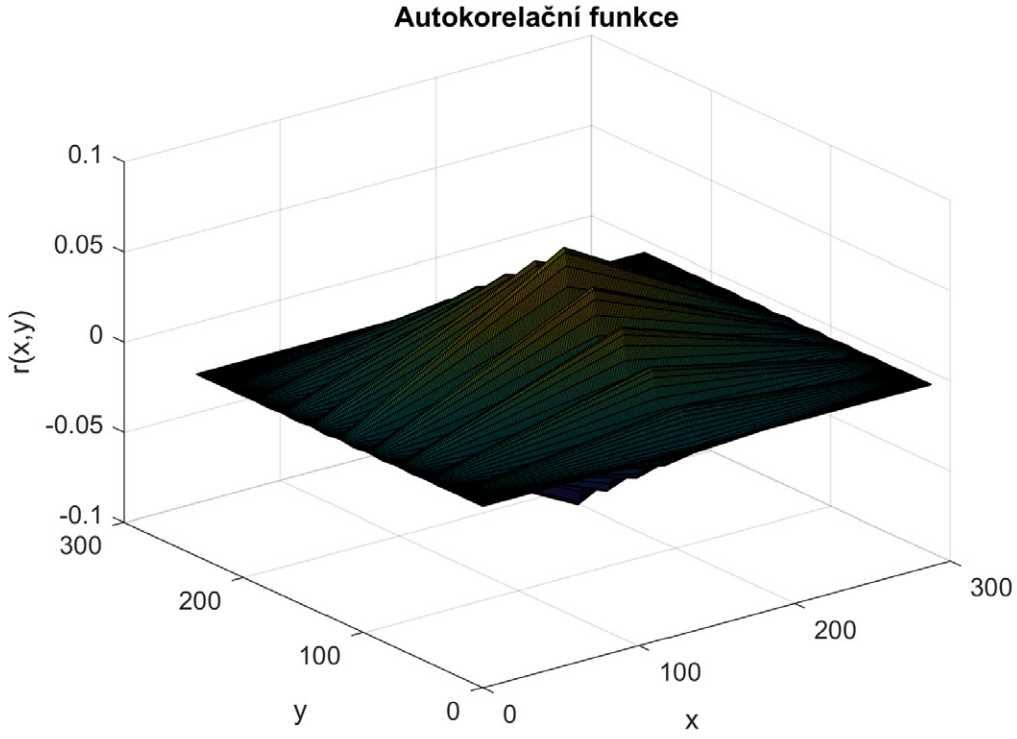
Úkol 2: Pomocí metody +/- průměrování odhadněte, zda byly obrazy mozku rušeny bílým šumem

```
% Definujte vahy +/- prumerovani:
M = 20;
weights = [1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1]/M;

% a) Pomoci autokorelacni funkce zjiste, zda jde o bily sum u brainsA:
restNoise = zeros(size(brainsA(:,:,1)));
for i=1:size(brainsA,3)
   restNoise = restNoise + brainsA(:,:,i)*weights(i);
end
```

```
% Prumer sumu:
mean(mean(restNoise))
% Prumer se blizi 0. Odpovida bilemu sumu.

% Aplikujte autokorelacni funkci na vektor ‚restNoise‘:
autocor = xcorr2(restNoise); % Obrazek 3A1
figure, surf(autocor) , title(‚Autokorelační funkce‘)
xlabel(‚x‘), ylabel(‚y‘), zlabel(‚r(x,y)‘)
% Autokorelacni funkce odpovida bilemu sumu.

% b) Pomoci autokorelacni funkce zjiste, zda jde o bily sum u brainsB:
restNoise = zeros(size(brainsB(:,:,1)));
for i=1:size(brainsB,3)
restNoise = restNoise + brainsB(:,:,i)*weights(i);
end

% Prumer sumu:
mean(mean(restNoise))
% Prumer se blizi 0. Odpovida bilemu sumu.

% Autokorelacni funkce:
autocor = xcorr2(restNoise); % Obrazek 3A2
figure, surf(autocor) , title(‚Autokorelační funkce‘)
xlabel(‚x‘), ylabel(‚y‘), zlabel(‚r(x,y)‘)
% Autokorelacni funkce neodpovida bilemu sumu. Obrazy byly ruseny ruzne
% posunutym sinusovym signalem.
```



Obrázek 3A1: Autokorelační funkce bílého šumu



Obrázek 3A2: Autokorelační funkce rušení různě posunutým sinusovým signálem

Příklad 3B: Kumulace obrázku Lenny

Aplikujte metodu kumulace s rovnoměrnými vahami a pevným oknem pro zvýraznění signálu v obrázku Lenny.

Úkol 1: Načtení dat

```
% Nactete obraz ‚data/Lenna.png‘:
A = imread(‚data/Lenna.png‘);

% Prevedte obraz do formatu double:
A = double(A);

% Sectete jednotlive barevne skaly:
Abw = A(:,:,1)+A(:,:,2)+A(:,:,3);

% Obraz znormalizujte a zobrazte pomocí funkce ‚imshow‘:
Abw = Abw-min(Abw(:));
Abw = Abw/max(Abw(:));
figure, imshow(Abw), title(‚Užitečná složka‘);
```

Úkol 2: Příprava dat pro kumulaci

```
% Vytvorte sadu M obrazku Leny s nahodnym rusenim (rovnomerne rozdeleny sum
% s nulovou stredni hodnotou).

% Definujte SNR a pocet repetic:
SNR = 1; % Napriklad 1
M = 100; % Napriklad 100

% Vytvorte matici rovnomerneho rozdeleneho sumu s hodnotami mezi -1 a 1 o
% velikosti (pocet pixelu sirka)×(pocet pixelu vyska)×M.
myNoise = (rand(512,512,M) - 0.5)*2;

% Pomoci SNR vypocitejte amplitudu sumu.
Asignal = 1; % Amplituda signalu je 1 diky predchozi provedene normalizaci
Anoise = Asignal/sqrt(SNR);

% Vynasobte sum jeho amplitudou ‚Anoise‘ a zobrazte si jeden obrazek sumu:
myNoise = Anoise*myNoise;
figure, imshow(myNoise(:,:,1)), title(‚Jedna repetice šumu‘);

% S vyuzitim matice ‚mynoise‘ vytvorte M ruzne zasumenych obrazu Lenny:
D = zeros(512,512,M);
for i = 1:M
   D(:,:,i)=Abw+myNoise(:,:,i);
end
figure, imshow(D(:,:,20)), title(‚Jedna repetice směsi‘); % Obrazek 3B1
```

Obrázek 3B1: Zašuměný obraz

Úkol 3: Očistěte obraz od šumu pomocí kumulace s rovnoměrnými vahami

```
kA = zeros(512,512);
weight=1/M;

for i=1:M
  kA = kA + D(:,:,i)*weight;
end

% Zobrazte obrazek Lenny po kumulaci:
figure, imshow(kA); title('Restaurovaný obraz po kumulaci'); % Obrazek 3B2
```

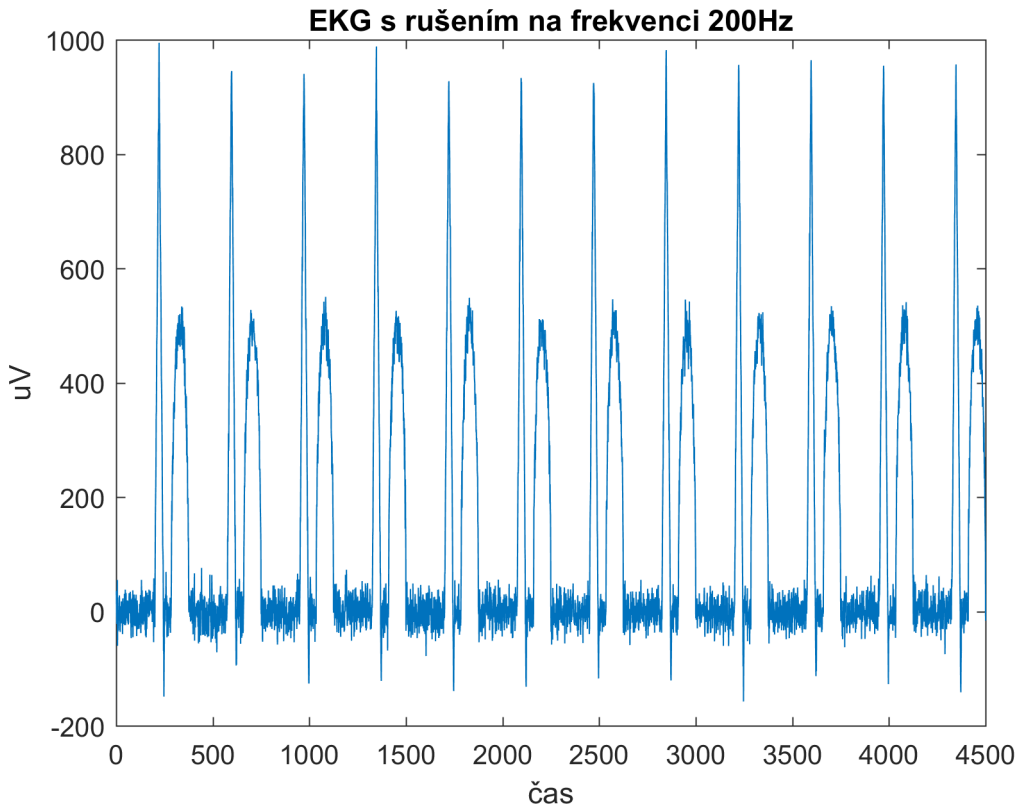
Obrázek 3B2: Restaurovaný obraz

Příklad 3C: Detekce repetic v EKG

Najděte v měření EKG repetice pomocí autokorelační funkce a detekce maxima a zvýrazněte pomocí kumulace s pevným oknem poměr signálu k šumu. Repetice jsou v tomto příkladu stejně dlouhé, není tedy potřeba žádného algoritmu pro zarovnání repetic.

Úkol 1: Načtení dat

```
% Nactete data k prikladu z \data\ekgSum.mat:
% Data byla vymodelovana pomoci nastroje:
% https://www.physionet.org/physiotools/matlab/ECGwaveGen/
load(char(strcat(pwd, ‚\data\ekgSum.mat‘)));

% Podle grafu definujte pocet repetic ‚n‘.
plot(ecgSum) % Obrazek 3C1
xlabel(‚čas‘), ylabel(‚uV‘), title(‚EKG s rušením na frekvenci 200Hz‘)
n=12;
```



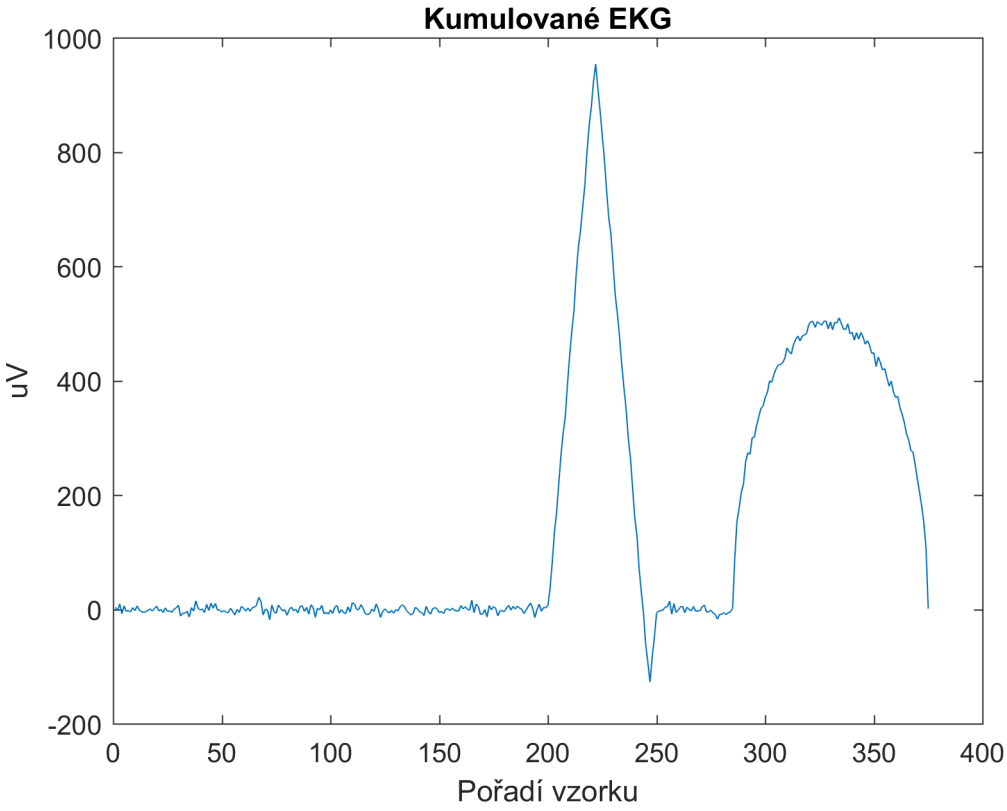
Obrázek 3C1: EKG s rušením

Úkol 2: Autokorelační funkce

```
% Vypocitejte autokorelacni funkci:
autocor = xcorr(ecgSum);
% Autokorelacni funkce je symetricka, staci polovina:
autocorHalf = autocor(ceil(length(xcorr(ecgSum))/2):end);
```

Úkol 3: Detekce repetic

```
% Najdete maxima v autokorelacni funkci. Predpokladame 12 repetic.
% Tj. napriklad naleznete jedno maximum a dale hledejte druhe maximum,
% ktere nebude v okoli +100 bodu, aby nedoslo k nalezeni bodu vedle
% prvniho maxima:
index = 1;
maximum = zeros(n,1);
for i = 1:n
   [~,pos] = max(autocorHalf(index:end));
   pos = pos + index - 1;
   maximum(i) = pos;
   index=pos+100;
end

% Nalezena maxima pouzijte k rozrezani puvodniho signalu na repetice.
% Ulozte je do promenne ‚repetition‘:
repetition = cell(1,12);
for i=1:n-1
   repetition{i} = ecgSum(maximum(i):maximum(i+1)-1);
end
repetition{12} = ecgSum(maximum(12):end);

% Zprumerujte repetice a zobrazte vyhlazenou repetici. Predpokladejme,
% ze dopredu nezname pocet vzorku.
cumulation = zeros(1, min(diff(maximum)));
for i = 1:n
   helper = repetition{i};
   helper = helper(1:length(cumulation));
   cumulation = cumulation + helper;
end
cumulation = cumulation/n;
plot(cumulation) % Obrazek 3C2
title(‚Kumulované EKG‘), xlabel(‚Pořadí vzorku‘), ylabel(‚uV‘)
```



Obrázek 3C2: EKG po kumulaci

Příklad 4A: Ergodicita a stacionarita časových řad

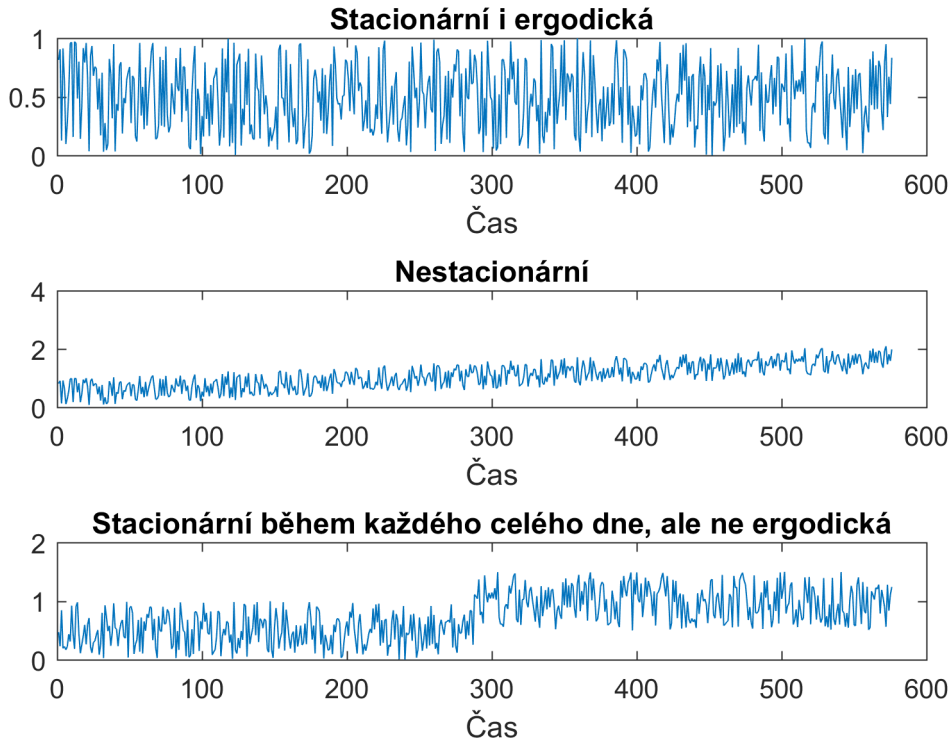
Načtěte data ze složky data\casove_rady se třemi časovými řadami a podle grafu určete, zda byly generovány stacionárními nebo ergodickými náhodnými procesy. Časové řady jsou nějaká měření prováděná v 5 minutových intervalech během dvou po sobě jdoucích dnech.

Úkol 1: Načtení dat

```
load(char(strcat(pwd, ‚\data\casove_rady\casovaRada1.mat‘)));
load(char(strcat(pwd, ‚\data\casove_rady\casovaRada2.mat‘)));
load(char(strcat(pwd, ‚\data\casove_rady\casovaRada3.mat‘)));
```

Úkol 2: Vykreslete a okomentujte, zda jde o časové řady stacionární nebo ergodické

```
subplot(3,1,1) % Obrazek 4A1
plot(timeSeries1)
title(‚Stacionární i ergodická‘)
xlabel(‚Čas‘)

subplot(3,1,2)
plot(timeSeries2)
title(‚Nestacionární‘) % Ma trend, meni se statisticke vlastnosti
xlabel(‚Čas‘)

subplot(3,1,3)
plot(timeSeries3)
title(‚Stacionární během každého celého dne, ale ne ergodická‘)
xlabel(‚Čas‘)
% Behem dne se nemeni statisticke vlastnosti.
% Nelze odhadnout statisticke vlastnosti libovolnym merenim, casova rada
% tedy nemuze byt ergodicka.
```



Obrázek 4A1: Stacionarita a ergodicita časových řad

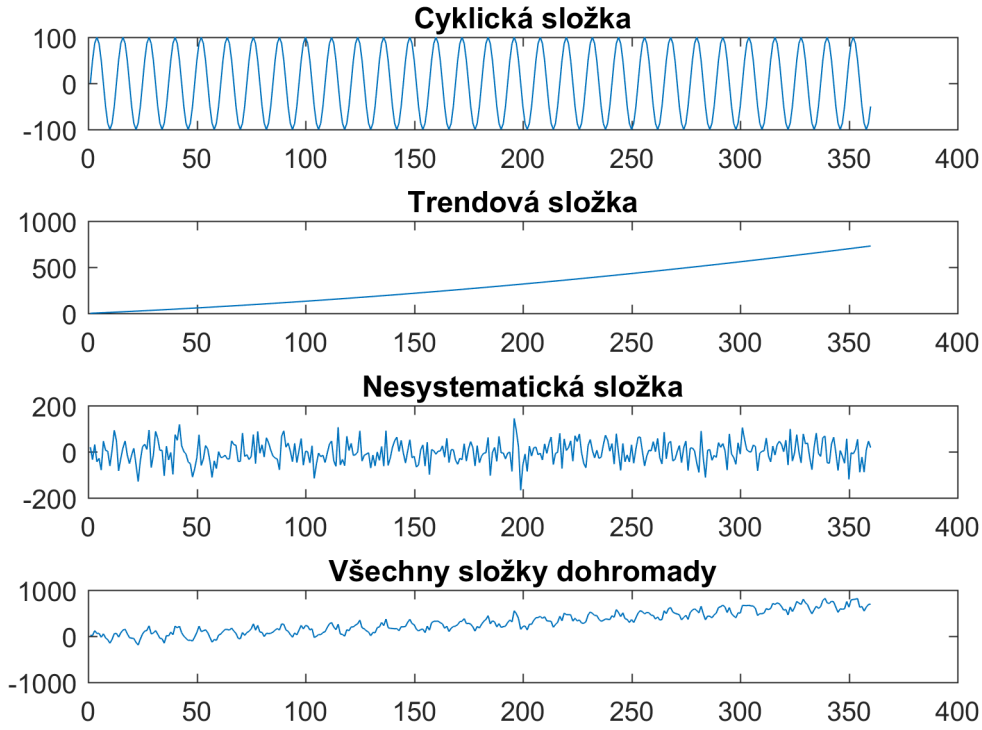
Příklad 4B: Dekompozice časových řad 1

Modelujte jednotlivé složky časové řady (trendovou, cyklickou a nesystematickou) a sestavte z nich časovou řadu. Pomocí regrese odstraňte trendovou složku. Cyklickou složku odstraňte pomocí průměru periodicky opakujících se měření.

Úkol 1: Sestavení časové řady pomocí jednotlivých komponent

```
% Pomoci trendove, cyklicke a nesystematicke slozky seskladejte casovou radu:

% Vytvorte cyklickou slozku, a to pomoci funkce sinus s frekvenci 12 vzorku
% v periode. Predstavujme si, ze jedna perioda odpovida prubeznemu mereni
% po dobu jednoho roku, pricemz mereni probiha jednou za mesic. Modelujte
% 360 po sobe jdoucich mereni, tedy 30let.
t = 0:pi/6:pi*60-pi/6;
seasonalComponent = 100*sin(t);
subplot(4,1,1) % Obrazek 4B1
plot(seasonalComponent), title('Cyklická složka')

% Vytvorte slozku s kvadratickym trendem:
trendComponent = 0.01*t.^2+2*t+1;
subplot(4,1,2)
plot(trendComponent), title('Trendová složka')
```

```
% Vytvorte nesystematickou slozku pomoci funkce ‚randn‘:
irregularComponent = 50*randn(1,length(t));
subplot(4,1,3)
plot(irregularComponent), title(‚Nesystematická složka‘)

% Z pripravenych slozek vytvorte aditivni model:
timeSeries = trendComponent + seasonalComponent + irregularComponent;
subplot(4,1,4)
plot(timeSeries), title(‚Všechny složky dohromady‘)
```

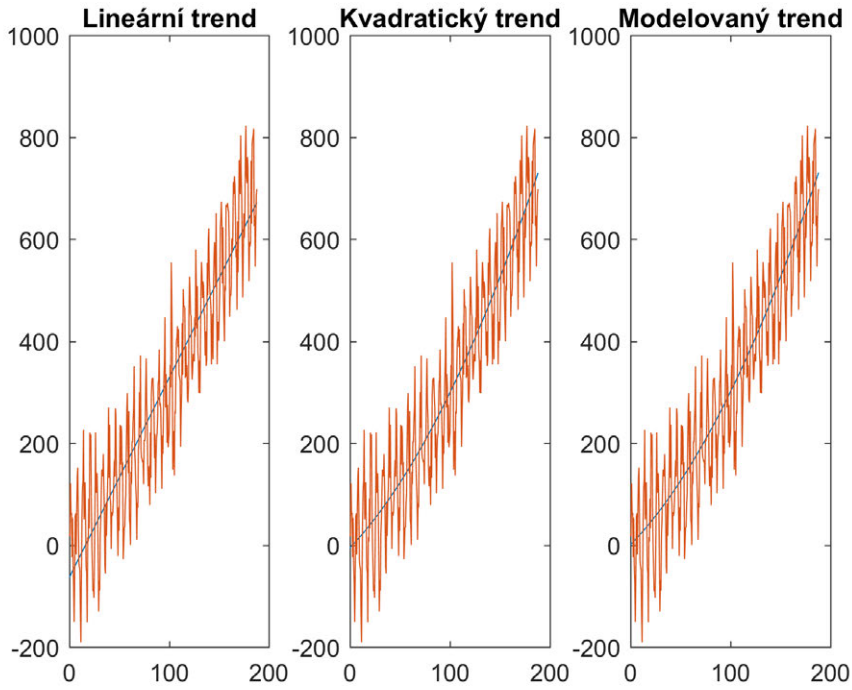


Obrázek 4B1: Složky časové řady

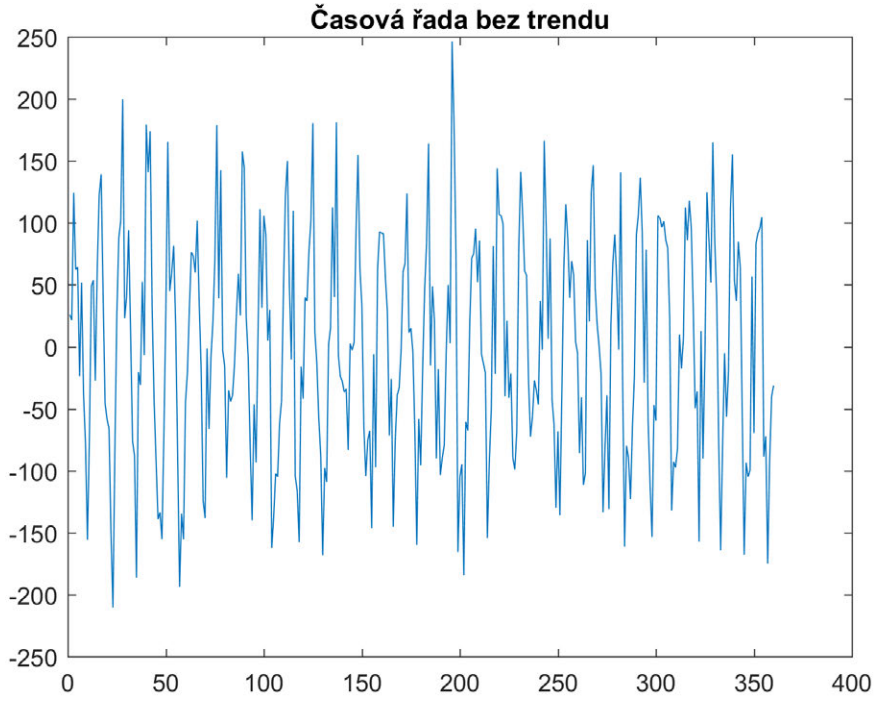
Úkol 2: Dekompozice časové řady

```
% Rozlozte casovou radu

% Odečtěte trendovou složku pomoci linearni i kvadraticke regrese:

% Linearni trend:
linTrendCoef = polyfit(t,timeSeries,1);
linTrend = polyval(linTrendCoef,t);

% Kvadraticky trend:
quadrTrendCoef = polyfit(t,timeSeries,2);
quadrTrend = polyval(quadrTrendCoef,t);
% Srovnejte oba modely s „realnou“ trendovou slozkou:
```

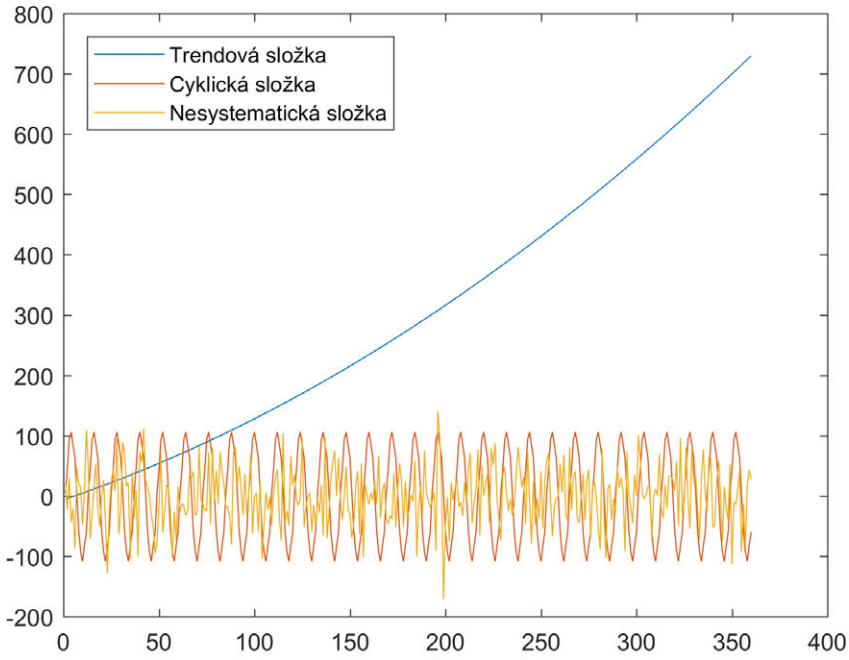
```
figure
subplot(1,3,1) % Obrazek 4B2
plot(t, linTrend), title(‚Lineární trend‘), hold on, plot(t,timeSeries)
subplot(1,3,2)
plot(t, quadrTrend), title(‚Kvadratický trend‘), hold on, plot(t,timeSeries)
subplot(1,3,3)
plot(t, trendComponent), title(‚Modelovaný trend‘), hold on, plot(t,timeSeries)

% Vyberte lepsi model trendu a ulozte do promenne ‚trendComponentModel‘:
trendComponentModel = quadrTrend;
% Vybereme model pro kvadraticky trend, casova rada s casem roste prudceji.
% Pro vyhodnoceni lze pouzit nastroje zname ze statistiky jako je napr.
% koeficient determinace (R2). Pro ucely prikladu nam postaci okometricke
% posouzeni.

% Odectete modelovanou trendovou slozku od casove rady:
timeSeriesTrendOut= timeSeries - trendComponentModel;
figure, plot(timeSeriesTrendOut), title(‚Časová řada bez trendu‘) % Obrazek 4B3

% Odectete cyklickou slozku. Spocitejte prumerne hodnoty pro jednolive
% „mesice“ a ulozte do promenne ‚monthAvg‘:
monthsReshape = reshape(timeSeriesTrendOut,12,length(timeSeriesTrendOut)/12);
monthAvg = mean(monthsReshape,2);

% „Mesicni“ prumery odectete od odpovidajicich mesicnich hodnot
% detrendovane casove rady:
seasonalComponentModel = repmat(monthAvg,length(timeSeriesTrendOut)/12,1)‘;
irregularComponent = timeSeriesTrendOut - seasonalComponentModel;
% Zbyla jen nesystemacitka slozka. Ta se da take modelovat napr. pomoci
% Boxovych-Jenkinsonovych modelu.

% Vykreslete modely pro slozky casove rady a zbylou nesystematickou slozku:
figure
plot(trendComponentModel), hold on % Obrazek 4B4
plot(seasonalComponentModel)
plot(irregularComponent)
legend({‚Trendová složka‘,‘Cyklická složka‘,‘Nesystematická složka‘}, ‚location‘, ‚northwest‘)
```



Obrázek 4B2: Model trendové složky



Obrázek 4B3: Časová řada po odstranění trendu



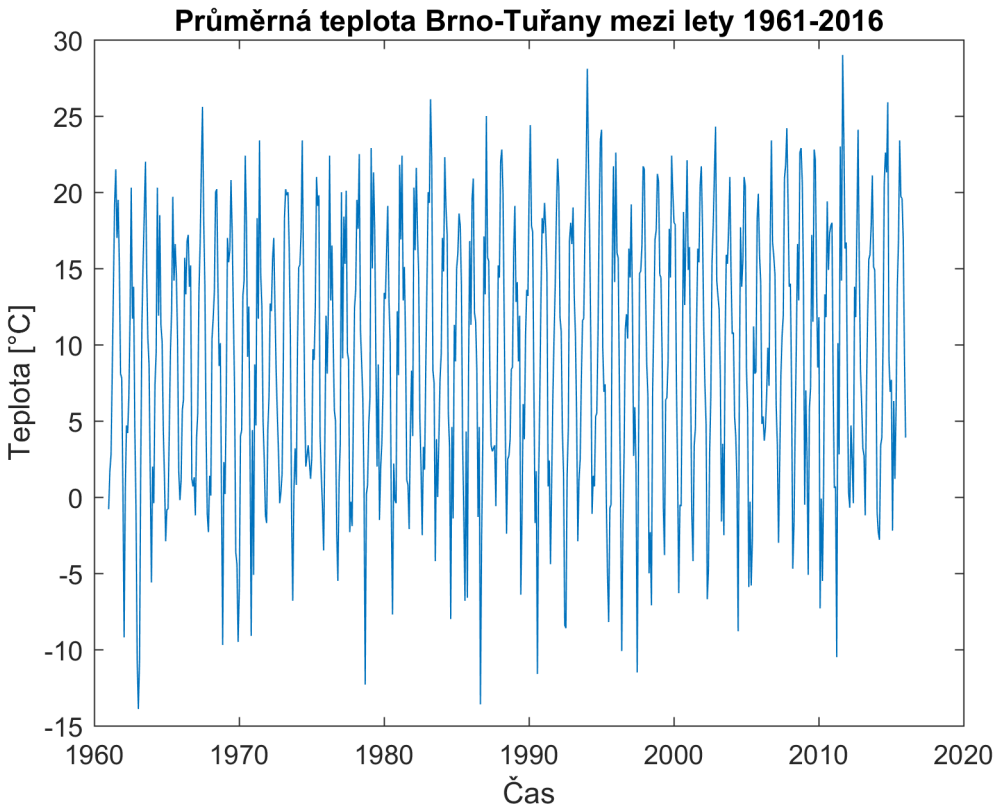
Obrázek 4B4: Modelované složky časové řady

Příklad 4C: Dekompozice časových řad 2

V Brně-Tuřanech se měří od roku 1961 teplota ovzduší. Datový soubor obsahuje měření průměrné teploty prvního dne v měsíci od roku 1961 do roku 2016. Cílem příkladu je dekomponovat časovou řadu, odhadnout trend vývoje teploty za využití klouzavého průměru a sezonní složku pomocí Butterworthova filtru.

Úkol 1: Načtení dat

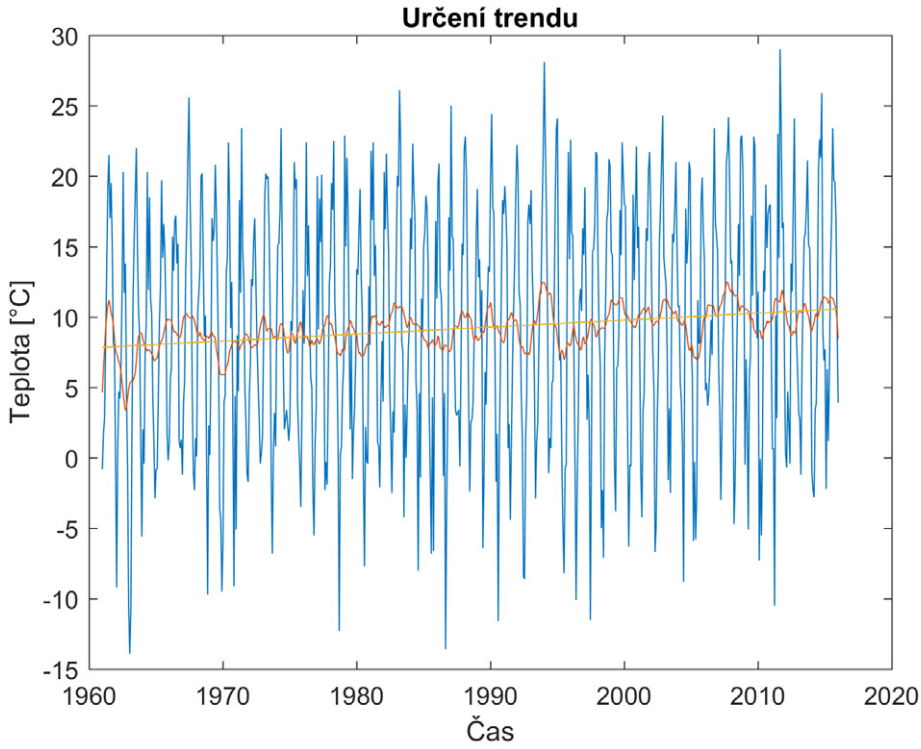
```
% Nactete soubor \data\prumerna_teplota_cr.csv a ulozte do promenne
% ‚temperature‘ a data si prohlednete. Popisky osy x budou roky ulozene
% v promenne ‚time‘:
%
% Zdroj dat: http://portal.chmi.cz/historicka-data/pocasi/denni-data#

temperature = csvread(char(strcat(pwd, ‚\data\prumerna_teplota_cr.csv‘)));
time = linspace(1961,2016, length(temperature));
plot(time, temperature) % Obrazek 4C1
title(‚Průměrná teplota Brno-Tuřany mezi lety 1961-2016‘)
xlabel(‚Čas‘), ylabel(‚Teplota [°C]‘)
```



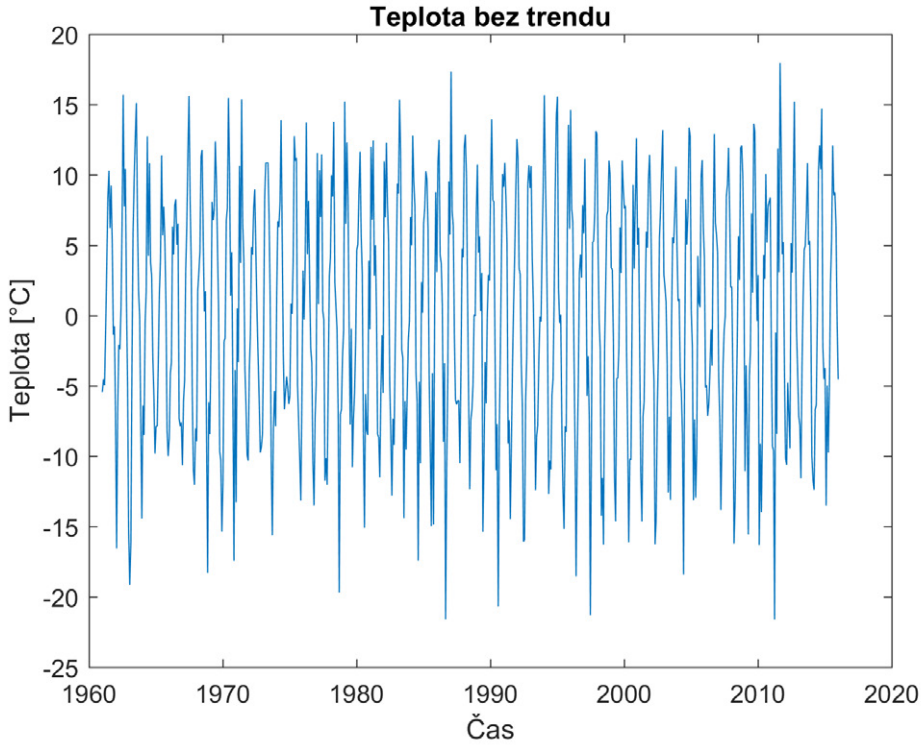
Obrázek 4C1: Časová řada teplot

Úkol 2: Odstranění trendové složky

```
% Pouzijte klouzavy prumer pro urceni trendove slozky a aplikujte ho
% na prumerne teploty:
h = [1/24; repmat(1/12,11,1); 1/24];
trend = conv(temperature,h,'same');
hold on, plot(time, trend), title('Určení trendu')
xlabel('Čas'), ylabel('Teplota [°C]')

% Srovnejte urceny trend pomoci klouzaveho prumeru a pomoci linearni
% regrese:
linTrendCoef = polyfit(time',temperature,1);
linTrend = polyval(linTrendCoef,time);
hold on, plot(time, linTrend) % Obrazek 4C2

% Do nove promenne ulozte casovou radu ocistenou o trendovou slozku:
temperatureTrendOut = temperature - trend;
figure, plot(time, temperatureTrendOut) % Obrazek 4C3
title('Teplota bez trendu'), xlabel('Čas'), ylabel('Teplota [°C]')
```



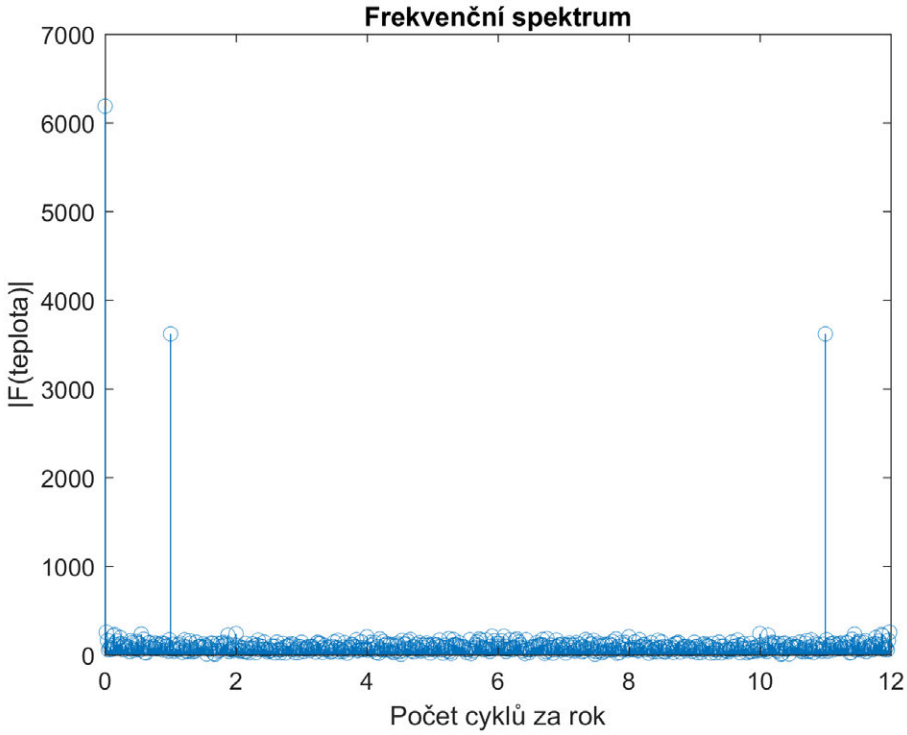
Obrázek 4C2: Určení trendové složky



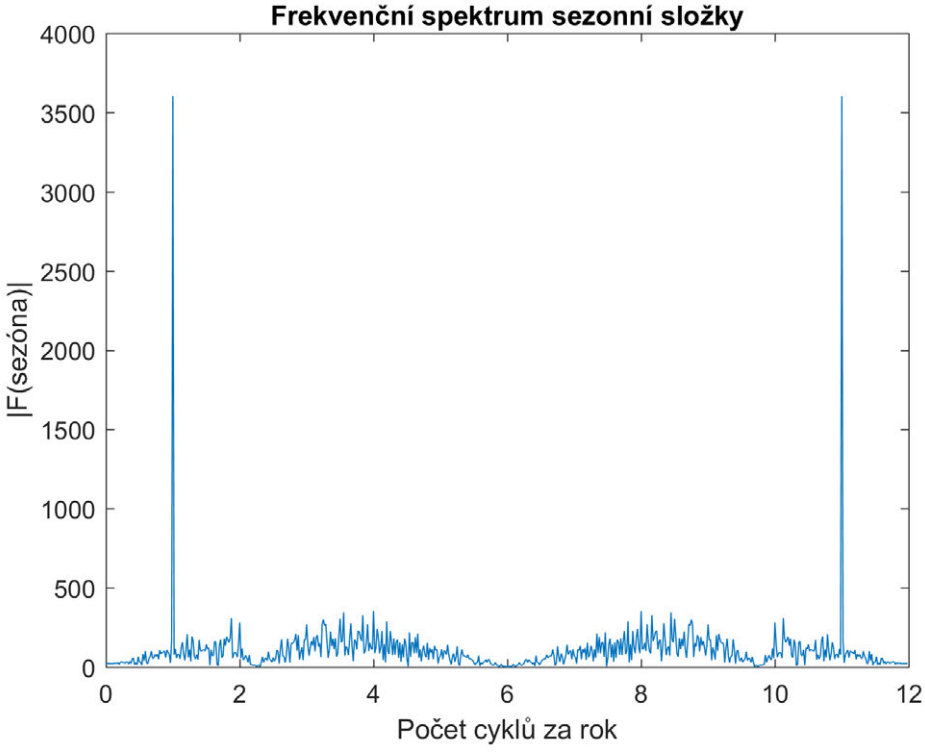
Obrázek 4C3: Teplota bez trendu

Úkol 3: Odstranění sezonní složky pomocí Butterworthova filtru

```
% Prohlednete si spektrum casove rady ocistene od trendove slozky
FS = 12; % Vzorkovani 12 vzorku za rok (1 cyklus...rok, 1 vzorek...mesic)
temperatureSpectrum = fft(temperature);
lenSpe = length(temperatureSpectrum);
x = 0:FS/lenSpe:(FS-FS/lenSpe);
stem(x,abs(fft(temperature))) % Obrazek 4C4
title(‚Frekvenční spektrum‘)
xlabel(‚Počet cyklů za rok‘), ylabel(‚|F(teplota)|‘);

% Ze spektra i logiky veci vime, ze sezonni slozka ma frekvenci kolem 1Hz.
% Navrhnete Butterworthuv filtr typu horni propust, ktery bude tuto slozku
% eliminovat. Sledujte charakteristiky filtru:
RAD = 10;
CUTOFF = 2;
highIIR = fdesign.highpass(‚N,F3db‘,RAD,CUTOFF,FS);
butterHighIIR = design(highIIR,‘butter‘);
fvtool(butterHighIIR) % Filtr s hranicni frekvenci 2Hz tlumi signal na frekvenci 1Hz.
% Pozn.: alternativne lze pouzit funkce ‚butter‘

% Pomoci filtru eliminujte sezonni slozku. Zkontrolujte, ze v jejim spektru
% chybi harmonicke slozky nizsi nez 2Hz.
irregularComponent = filter(butterHighIIR, temperatureTrendOut);
figure, plot(time, irregularComponent)
title(‚Nesystematická složka‘), xlabel(‚Čas‘), ylabel(‚Teplota [°C]‘)
figure, plot(x,abs(fft(irregularComponent)))
title(‚Frekvenční spektrum nesystematické složky‘), xlabel(‚Počet cyklů za rok‘),
ylabel(‚|F(rušení)|‘)

% Ziskejte sezonni slozku a opet si prohlednete jeji spektrum:
seasonalComponent = temperatureTrendOut-irregularComponent;
figure, plot(time, seasonalComponent)
title(‚Sezonní složka‘), xlabel(‚Čas‘), ylabel(‚Teplota [°C]‘)
figure, plot(x,abs(fft(seasonalComponent)))
title(‚Frekvenční spektrum sezonní složky‘) % Obrazek 4C5
xlabel(‚Počet cyklů za rok‘), ylabel(‚|F(sezóna)|‘)
```



Obrázek 4C4: Odhalení cyklické složky



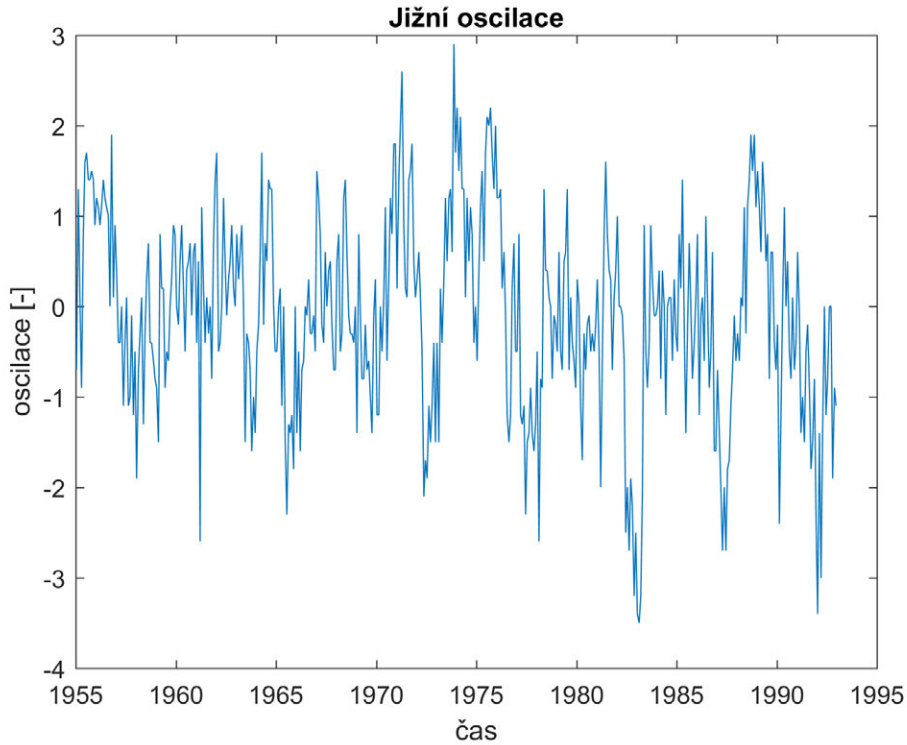
Obrázek 4C5: Spektrum modelované cyklické složky

Příklad 4D: Exponenciální vyhlazování a predikce

Jižní oscilace je definována jako barometrický tlakový rozdíl mezi Tahiti a Darwinovými ostrovy na hladině moře. Jižní oscilace může předpovědět El Niño, který ovlivňuje celosvětové počasí. Opakované hodnoty jižní oscilace menší než -1 typicky definují El Niño. Načtěte časovou řadu jižních oscilací za 40 let (vzorkovací perioda je jeden měsíc) a pomocí techniky exponenciálního vyhlazování a predikce vypočítejte predikce El Niño na jeden měsíc dopředu. Pro realizaci použijte exponenciální filtr a experimentujte s jeho řádem a koeficientem zapomínání při definici jeho vah.

Úkol 1: Načtěte data a zobrazte časový průběh jižních oscilací

```
data = dlmread('data/elnino.csv',';');
oscillation = data(:,1);
years = data(:,2);
plot(years, oscillation) % Obrazek 4D1
title('Jižní oscilace'), xlabel('čas'), ylabel('oscilace [-]');
```



Obrázek 4D1: Jižní oscilace

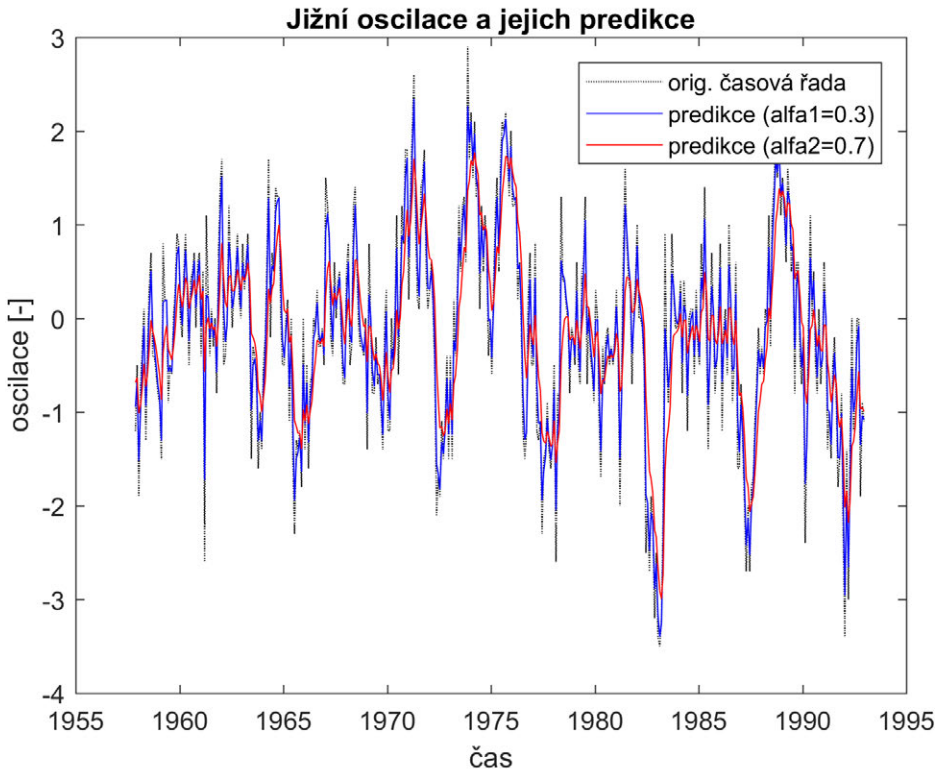
Úkol 2: Volte různé nastavení predikčního filtru a spočítejte hodnoty jeho impulsní charakteristiky

```
% Zvolte rad filtru:
M = 35;

% Stanovte a vyzkousejte 2 ruzne koeficienty zapominani (parametr alfa):
ALFA1 = 0.3;
ALFA2 = 0.7;
% Stanovte impulzni charakteristiku predikcniho filtru pro oba koeficienty
% zapominani:
h1 = zeros(1,M);
h2 = zeros(1,M);
for i=1:M
   h1(i)=ALFA1^(i-1); % Nastaveni zapominani ve vahach exponencialniho filtru
   h2(i)=ALFA2^(i-1);
end
h1 = h1/sum(h1); % Normalizace, aby filtr nezesiloval ani nezeslaboval
h2 = h2/sum(h2);

% Pozn.: Alternativni zpusob vypoctu vah. Neni zde potreba resit
% normalizaci:
% for i=1:M
%    h1(i) = ALFA1*(1-ALFA1)^(i-1);
%    h2(i) = ALFA2*(1-ALFA2)^(i-1);
% end

% Aplikujte oba predikcni filtry na data:
oscillationPredict1 = conv(oscillation,h1,'valid');
oscillationPredict2 = conv(oscillation,h2,'valid');
% Pozn.: Vektor predikovanych hodnot je kratsi nez vstupni casova rada kvuli okrajovemu
jevu konvoluce.
```

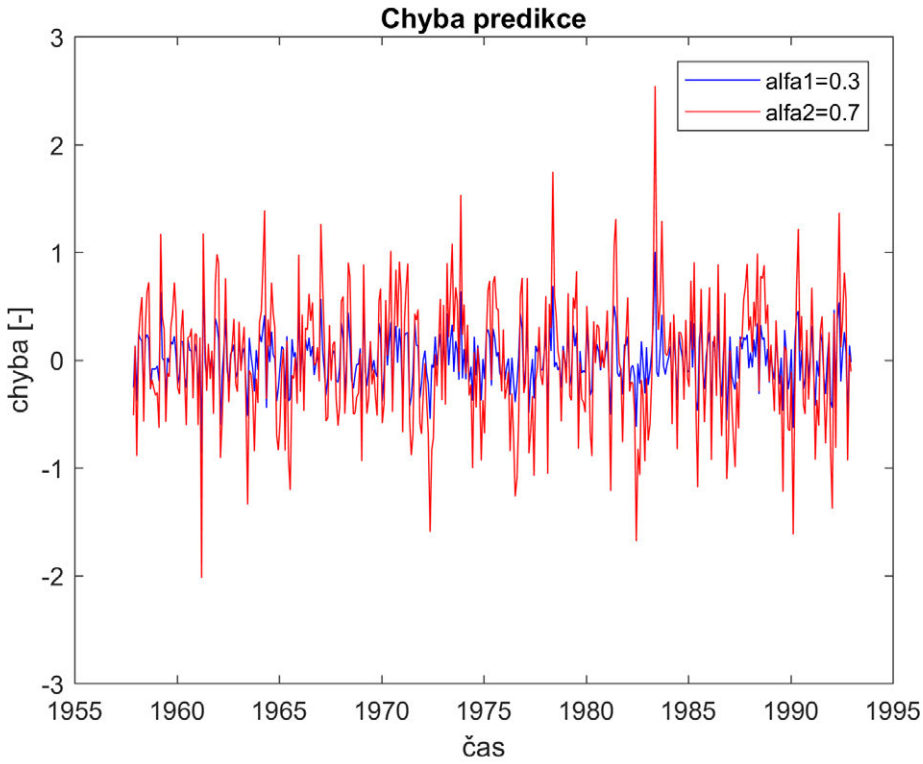
Úkol 3: Zobrazte výsledné predikce a vypočítejte chybu predikce s horizontem predikce 1 měsíc

```
% Orezte vektor s oscilacemi a vektor s roky tak, aby jejich vzorky casove
% odpovidaly vzorkum vektoru s predikcemi:
oscillationCut = oscillation(M:length(oscillationPredict1)+M-1); % Prvni vzorek vstupni
casove rady souvisejici..
yearsCut=years(M:length(oscillationPredict1)+M-1); % ..s prvnim validnim vzorkem na
vystupu predikcniho filtru je ten Mty.

% Vypocitejte chyby obou predikci:
e1 = oscillationCut-oscillationPredict1;
e2 = oscillationCut-oscillationPredict2;

% V jednom grafu zobrazte puvodni casovou radu s oscilacemi a obe predikce:
figure
plot(yearsCut,oscillationCut,'k:'), hold on; % Obrazek 4D2
```

```
plot(yearsCut,oscillationPredict1,'b');
plot(yearsCut,oscillationPredict2,'r');
title('Jižní oscilace a jejich predikce');
xlabel('čas'), ylabel('oscilace [-]');
legend('orig. časová řada', ['predikce (alfa1=' num2str(ALFA1) ')'], ['predikce (alfa2='
num2str(ALFA2) ')']);

% V jednom grafu zobrazte chybu obou predikci:
figure
plot(yearsCut,e1,'b'), hold on; % Obrazek 4D3
plot(yearsCut,e2,'r');
title('Chyba predikce');
xlabel('čas'), ylabel('chyba [-]');
legend(['alfa1=' num2str(ALFA1)], ['alfa2=' num2str(ALFA2)]);

% Na zaver vypocitejte stredni kvadratickou chybu (MSE) predikce a
% zhodnotte, ktera z nich je podle tohoto kriteria lepsi:
disp(sum(e1.^2)/length(e1)) % Koeficient alfa=0.3 je lepsi pro predikci nez alfa=0.7, MSE je
nekolikanasobne mensi.
disp(sum(e2.^2)/length(e2))
```



Obrázek 4D2: Predikce pomocí exponenciálního vyhlazování



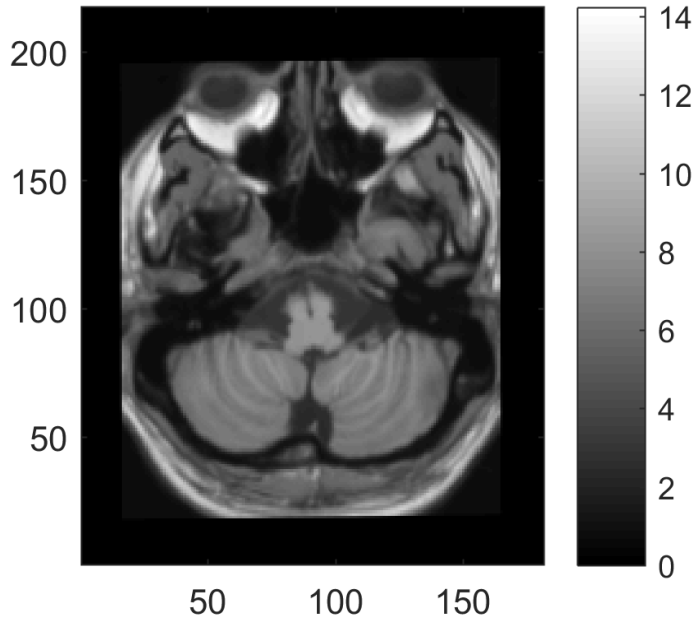
Obrázek 4D3: Chyby predikcí pomocí exponenciálního vyhlazování

Příklad 4E: Vzájemná informace

Vytvořte funkci pro výpočet sdruženého histogramu dvou náhodných proměnných a využijte ji pro výpočet vzájemné informace dvou řezů obrazem mozku A a B.

Úkol 1: Načtení dat a příprava dat

```
% Nahrat predpripravena data z ‚data\registrace.mat‘:
% A: Sablona z digitalniho atlasu mozku ICBM
% B: Obraz jednoho subjektu - T1 vazeny
load data\registrace.mat

% Zvolena souradnice na axialni ose:
Z = 25;
sliceA = A.img(:,:,Z);
sliceB = B.img(:,:,Z);

% Zobrazeni v souradnem systemu xy:
figure, imagesc(flipud(rot90(sliceA))); % Obrazek 4E1
colormap(gray); colorbar; truesize; axis xy;

figure, imagesc(flipud(rot90(sliceB)));
colormap(gray); colorbar; truesize; axis xy;
```



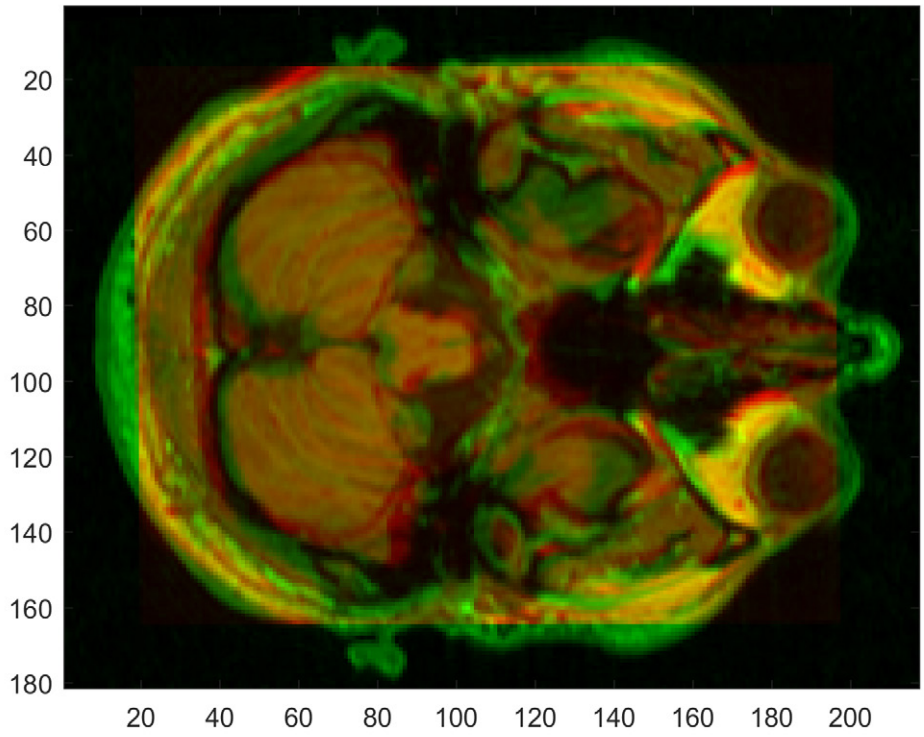
Obrázek 4E1: Řez hlavou

Úkol 2: Výpočet sdruženého histogramu

```
cd(‚funkce‘);

% Doplnte funkci ‚jointhist‘:
jointHist=jointhist(sliceA,sliceB,128,2);

% Normalizujte sdruzeny histogram:
N=sum(jointHist(:));
jointHistN = jointHist/N;

% Vypocitejte marginalni distribuce:
m2=sum(jointHistN,1);
m1=sum(jointHistN,2);
```

Úkol 3: Výpočet vzájemné informace

```
% Doplnte funkci ‚mutualinfo‘:
mutualInfo = mutualinfo(m1,m2,jointHistN);
disp(mutualInfo)
% Hodnota vzajemne informace bude nejvyssi, pokud se pro sliceA a sliceB vyberou rezy se
% stejnou z-ovou souradnici. Pokud zvolime pro sliceB souradnici jinou (napr. z+10 nebo
z-20),
% mela by vzajemna informace vyjit nizsi.

% Okometricke posouzeni kvality licovani:
channels(sliceA,sliceB); % Obrazek 4E2
```



Obrázek 4E2: Zobrazení výsledku slícování

Funkce: jointhist

```
function JH=jointhist(V1,V2,urovne,prah)
% jointhist - Vypocet sdruzeneho histogramu (empiricke sdruzene hustoty
%         pravdepodobnosti) dvou nahodnych promennych V1 a V2.
%
% urovne: pocet urovni sedi
% prah: nejmensi hodnota sedotonove urovne, ktera neni pozadi
%
% JH=jointhist(V1,V2,urovne,prah)

V1=V1(:); V1=V1-min(V1); V1=round( (V1/max(V1)) * (urovne-1) );
V2=V2(:); V2=V2-min(V2); V2=round( (V2/max(V2)) * (urovne-1) );

JH=zeros(urovne,urovne);

I=find(V1>prah & V2>prah);
for index=1:length(I)
  % Vlozte kod pro vypocet sdruzeneho histogramu.
  % Projdete vsechny nadprahove hodnoty obrazu V1, V2
  % a pro kazdy intenzitni par naakumulujte jeho cetnost.
  m=V1(I(index)); n=V2(I(index));
  JH(m+1,n+1)=JH(m+1,n+1)+1;
end
```

Funkce: mutualinfo

```
function MI=mutualinfo(marginal1,marginal2,joint)
% mutualinfo - Vypocita hodnotu vzajemne informace na zaklade empirickych
%          pravdepodobnostnich funkci dvou nahodnych promennych
%          (marginal1 a marginal2) a jejich sdruzene hustoty
%          pravdepodobnosti (joint).
%
% MI=mutualinfo(marginal1,marginal2,joint);


MI=0;
for m=1:length(marginal1)
  for n=1:length(marginal2)
    %
    % Doplnte vypocet MI.
    % Tzn. algoritmizujte vzorec vzajemne informace
    % (http://en.wikipedia.org/wiki/Mutual_information)
    %
    nominator=joint(m,n);
    denominator=marginal1(m)*marginal2(n);
    if nominator>0 && denominator>0
      MI=MI+nominator*log2(nominator/denominator);
    end
  end
end
```

Kapitola 5: Adaptivní zpracování a predikce časových řad

Příklad 5A: Optimální filtrace pro identifikaci neznámého systému

Určete, jakým filtrem byla vyhlazena časová řada měření teplot v České republice mezi lety 1961 a 2016.

Úkol 1: Načtení dat

```
% Nactete soubor \data\prumerna_teplota_cr.csv a ulozte do promenne
% ‚temperature‘, dale nactete soubor \data\prumerna_teplota_cr_vyhlazena.mat,
% kde je casova rada s teplotami vyhlazena neznamym filtrem.
% Zdroj dat: http://portal.chmi.cz/historicka-data/pocasi/denni-data#
temperature = csvread(char(strcat(pwd, ‚\data\prumerna_teplota_cr.csv‘)));
load(char(strcat(pwd, ‚\data\prumerna_teplota_cr_vyhlazena.mat‘)));

% Prohodte poradi mereni v obou casovych radach od posledniho k prvnimu
% vzorku (Duvod: vektor xN v odvozeni normalnich rovnic obsahuje casove prevracene
% poradi vzorku budici casove rady. Z takto prevraceneho vektoru se pocita i
% autokorelacni matice Rxx)
temperature = flip(temperature);
temperatureSmooth = flip(temperatureSmooth);
```

Úkol 2: Návrh filtru pomocí optimální filtrace

```
% Pro ruzne rady filtru (‚order‘) odhadnete impulzni charakteristiku:
for order = 2:7

% Pomoci funkce ‚corrmtx‘ vypocitejte autokorelacni matici a ulozte ji
% do matice ‚R‘:
[~,R] = corrmtx(temperature,order-1,‘autocorrelation‘);

% Pomoci funkce ‚xcorr‘ vypocitejte vektor krizovych korelaci mezi puvodni
% a vyhlazenou casovou radou:
p = xcorr(temperature,temperatureSmooth,order-1,‘biased‘);

% Autokorelacni funkce je symetricka kolem pocatku, pro vypocet potrebujeme
% pouze jeji pravou cast s kladnymi argumenty (kladna casova posunuti):
p = p(ceil(length(p)/2):end);

% Vypocitejte impulzni charakteristiku a sledujte, co se deje, pokud je rad
% predikcniho filtru zvolen vyssi, nez byl rad identifikovaneho systemu:
h=R\p;
display(h)
% Pozn.: Neznamy filtr mel impulzni charakteristiku [1/3 1/3 1/3]
% V situacich, kdy ma predikcni filtr vyssi rad nez identifikovany system
% (v tomto prikladu order>3), se vahy optimalniho filtru spravne nastavi
```

```
% na nulu (numericky velmi blizko nule), a tedy vice zpozdene vzorky vstupni posloupnosti
% se ve vypoctu vystupniho vzorku neuplatni. Naopak, je-li rad predikcniho filtru nizsi nez
% vykazuje identifikovany system, nastavi se vahy optimalniho filtru tak, aby byla
% v ramci omezenych moznosti alespon dosazena co nejnizsi stredni kvadraticka chyba
predikce.

end
```

Příklad 5B: LMS filtr pro identifikaci distortion efektu

Předpokládejte, že se vám líbil distortion efekt, který použili hudebníci pro úpravu jedné jejich skladby, a nyní jej chcete použít i pro úpravu svojí nahrávky. Přičemž máte k dispozici původní i upravenou skladbu.

Použijte LMS algoritmus pro identifikaci efektu typu distortion, který byl použit hudebníky pro úpravu jejich skladby, a aplikujte ho na jiný zvukový soubor.

Úkol 1: Načtení dat

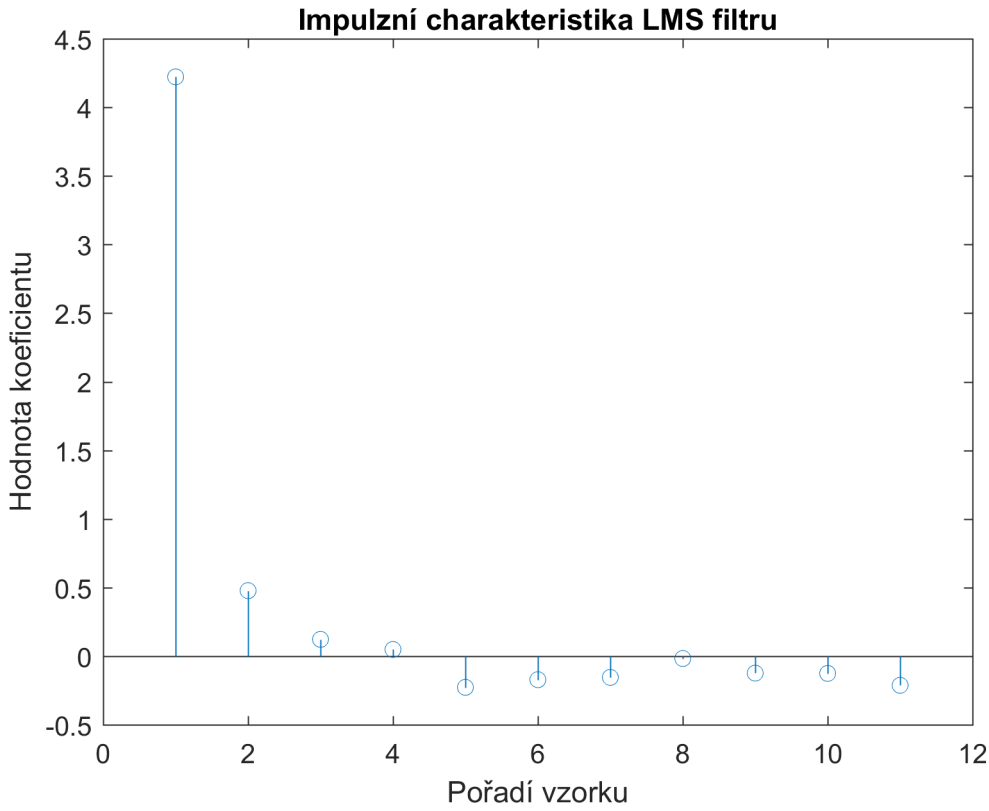
```
% Nactete soubor ‚\data\handel_distortion.wav' a soubor ‚handel.mat', ktery
% je soucasti datove sady v Matlabu:
[distortedSound,Fs] = audioread(char(strcat(pwd, ‚\data\handel_distortion.wav')));
load(‚handel');
```

Úkol 2: Definice LMS filtru

```
% Definujte LMS filtr pro aproximaci distortion efektu, kterym byl upraven
% nacteny soubor.

% Urcete pocet vzorku impulzni charakteristiky:
ORDER = 11; % Napr. 11

% Urcete delku kroku (alfa) LMS algoritmu:
ALFA = 0.01; % Napr. 0.01

% Definujte a aplikujte LMS filtr. Vytvorte objekt LMS filtru pomoci
% ‚dsp.LMSFilter' nebo pouzijte funkci lms, kterou najdete ve slozce
% ‚funkce/hayes':
lmsFilter = dsp.LMSFilter(ORDER, ‚StepSize', ALFA);
[yOutput,err,h] = lmsFilter(y,distortedSound);

% Postup pomoci funkce lms:
%
% Postupy vypoctu byly testovane v Matlabu verze 9.2.0.538062 (R2017a).
% Ve starsich verzich nemusi fungovat vypocet pomoci ‚dsp.LMSFilter', proto
% je zde uveden i alternativni vypocet impulzni charakteristiky LMS filtru
% pomoci funkce lms (autor: M.H. Hayes):
%
% cd(‚funkce/hayes')
% [A,err] = lms(y,distortedSound,ALFA,ORDER);
% err = err';
% h = A(end,:);
% yOutput = conv(y,h,'same');
```

Úkol 3: Grafy pro adaptaci LMS filtru a impulzní charakteristiku

```
% Vykreslete do jednoho grafu chybu, ocekavanou casovou radu a casovou radu
% modelovanou pomoci filtru:
plot(1:length(y), [distortedSound,yOutput,err]);
legend('Očekávaný výstup', 'Výstup', 'Chyba');
xlabel('Čas');

% Vykreslete impulzni charakteristiku filtru:
figure, stem(1:ORDER, h); % Obrazek 5B1
title('Impulzní charakteristika LMS filtru');
xlabel('Pořadí vzorku');
ylabel('Hodnota koeficientu');
```



Obrázek 5B1: Impulzní charakteristika filtru

Úkol 4: Stabilita LMS filtru

```
% Overte stabilitu filtru v casove domene:
sum(abs(h)) % Stabilni, neni v nekonecnu
```

Úkol 5: Použití LMS filtru

```
% Nactete do Matlabu libovolnou melodii a pouzijte na ni adaptovany
% distortion efekt:
```

```
% Priklad 1:
handelDist = conv(y,h); % Aplikace na puvodni signal
sound(handelDist)

% Priklad 2:
load gong
sound(y)
gongDist = conv(y,h);
sound(gongDist)
```

Příklad 5C: LMS filtr pro odstranění šumu ze zvukového záznamu

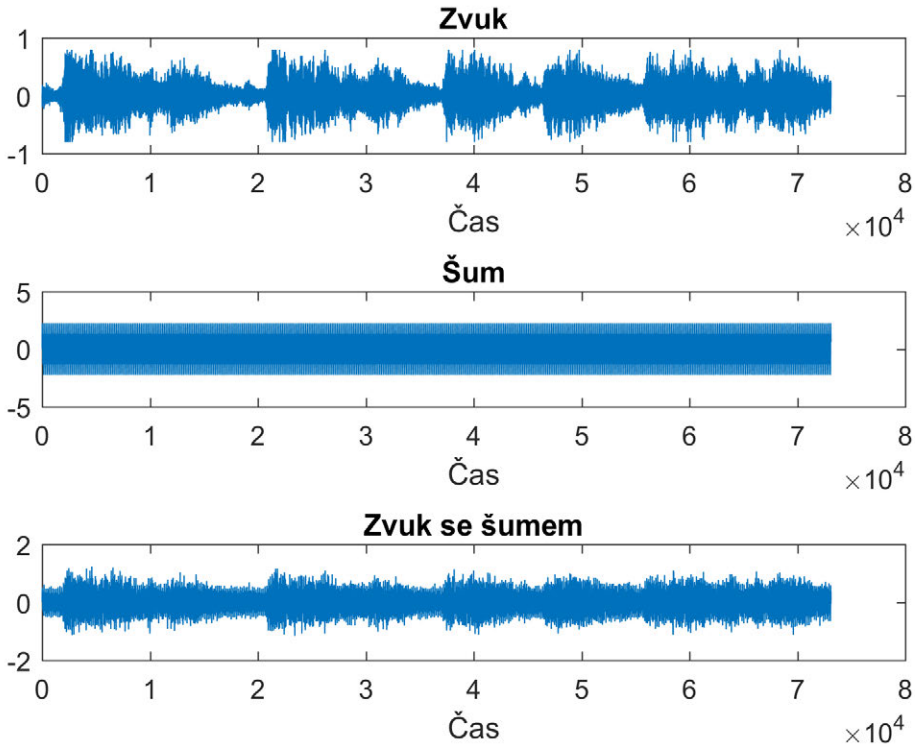
Předpokládejte, že jste na koncertě a máte za úkol pro organizátory pořídit nahrávku. V koncertní hale je ale rušivý signál. Pořídíte tedy před začátkem koncertu nahrávku tohoto šumu. Během koncertu nahrajete na diktafon záznam hudby. Protože chcete organizátorům předat kvalitní záznam bez rušení, musíte toto rušení eliminovat.

Použijte LMS algoritmus pro odstranění šumu.

Úkol 1: Načtení dat a vygenerování šumu

```
% Nactete zvukovy soubor ‚handel', ktery je soucasti matlabu:
load(‚handel');

% Generujte rusivou slozku jako smes tri signalu o frekvenci 50, 100 a
% 200Hz a prictete ji ke zvukove stope:

% Definujte vzorkovaci periodu. (Vzorkovaci frekvence se nacte spolu s
% nactenim zvukove stopy handel jako ‚Fs'):
Ts = 1/Fs;

% Definujte delku trvani sumu:
dt = 0:Ts:length(y)*Ts-Ts;

% Definujte frekvence sumu:
f1 = 50; f2 = 100; f3 = 200;

% Do promenne ‚noise' ulozte smes tri signalu. Amplitudu definujte jako
% A=1:
noise = sin(2*pi*f1*dt)'+sin(2*pi*f2*dt)'+sin(2*pi*f3*dt)';

% Spojte zvukovou stopu a sum, ten nejakym zpusobem deformujte napr.
% vynasobte koeficientem 0.2 nebo upravte nejakym filtrem. Vysledek si
% poslechnete a v maticovem grafu vykreslete hodnoty vsech tri signalu
% (zvuk, sum, zvuk+sum).
handelNoise = y + 0.2*noise;
sound(handelNoise)
```

```
% V maticovem grafu 3×1 zobrazte nacteny zvuk ‚handel', sum ‚noise', a zvuk
% smichany s sumem ‚handelNoise':
subplot(3,1,1) % Obrazek 4C1
plot(y), xlabel(‚Čas'), title(‚Zvuk')
subplot(3,1,2)
plot(noise), xlabel(‚Čas'), title(‚Šum')
subplot(3,1,3)
plot(handelNoise), xlabel(‚Čas'), title(‚Zvuk se šumem')
```



Obrázek 5C1: Zvuk se šumem

Úkol 2: Definice LMS filtru

```
% Vytvorte adaptivni filtr pomoci LMS algoritmu, do ktereho bude vstupovat
% rusiva slozka a ocekavany vystup bude smes uzitecne slozky a ruseni
% ‚handelNoise'. Nakonec Vas bude zajimat vektor s chybami v prubehu
% adaptace, ktery bude obsahovat zvukovy signal s postupnym odstranenim
% rusive slozky.

% Urcete pocet vzorku impulzni charakteristiky LMS filtru:
ORDER = 11; % Napr. 11

% Urcete delku kroku (alfa) LMS algoritmu:
ALFA = 0.01; % Napr. 0.01
```

```
% Definujte a aplikujte LMS filtr. Vytvorte objekt LMS filtru pomoci
% ‚dsp.LMSFilter' nebo pouzijte funkci ‚lms', kterou najdete ve slozce
% ‚funkce/hayes':
lmsFilter = dsp.LMSFilter('Length', ORDER, 'StepSize', ALFA);
[~,err,~] = lmsFilter(noise,handelNoise);

% Rezidua tvori cisty signal, poslechnete si vysledek:
sound(err)

% Pozn.: Proste odecteni ruseni nestaci, protoze sum je vynasoben
% koeficientem 0.2 (popripade filtrovan).
sound(handelNoise-noise)

% Postup pomoci funkce lms:
%
% Postupy vypoctu byly testovane v Matlabu verze 9.2.0.538062 (R2017a).
% Ve starsich verzich nemusi fungovat vypocet pomoci ‚dsp.LMSFilter', proto
% je zde uveden i alternativni vypocet impulzni charakteristiky LMS filtru
% pomoci funkce lms (autor: M.H. Hayes):
%
% cd('funkce/hayes')
% [~,err] = lms(noise,handelNoise,ALFA,ORDER);
% sound(err)
```

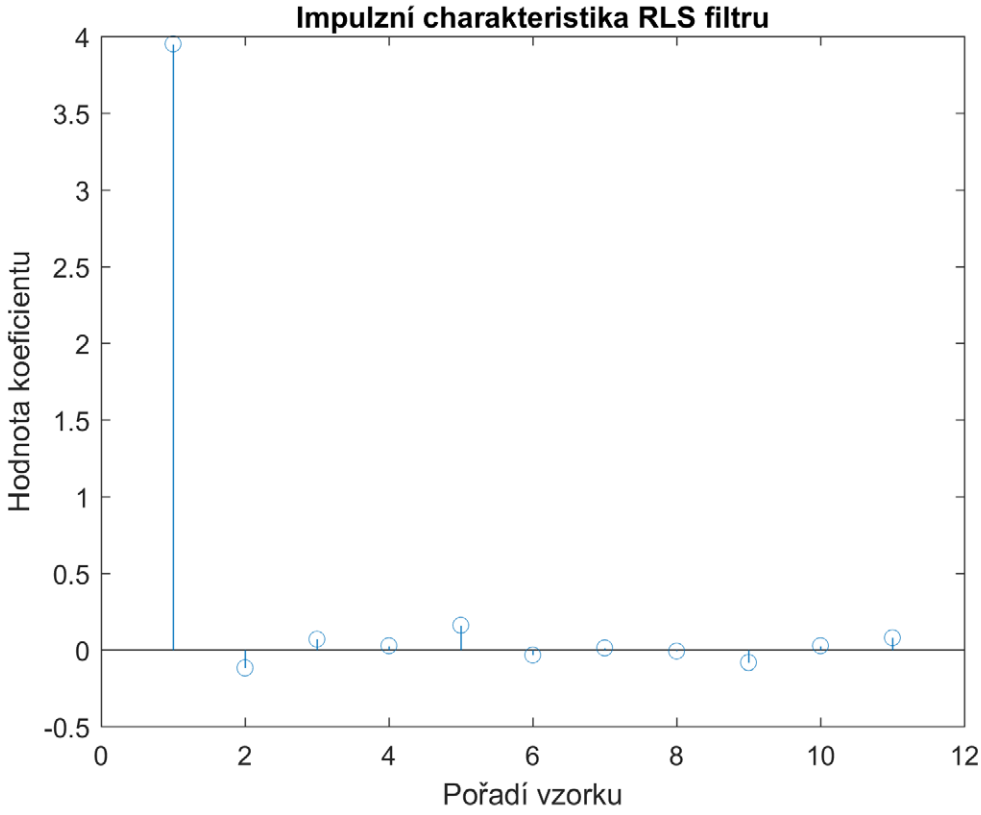
Příklad 5D: RLS algoritmus pro identifikaci systému a odstranění šumu

Obdobně jako v příkladu 5B a 5C identifikujte distortion efekt a vytvořte nástroj pro odstranění rušení ze souboru handel. Postupujte stejně jako v předchozích příkladech, ale místo LMS použijte RLS filtr.

Úkol 1: Identifikace distortion efektu

```
% Nactete soubor ‚\data\handel_distortion.wav' a nactete zvukovy soubor
% ‚handel':
[distortedSound,Fs] = audioread(char(strcat(pwd, '\data\handel_distortion.wav')));
load('handel');

% Urcete pocet vzorku impulzni charakteristiky hledaneho filtru:
ORDER = 11; % Napr. 11

% Urcete parametr lambda, ktery urcuje zapominani vlivu vzorku smerem
% do minulosti:
LAMBDA = 1; % Napr. 1

% Navrhnete RLS filtr a adaptujte ho tak, aby soubor handel byl vstupem a
% vektor ‚distortedSound' byl ocekavanym vystupem. Ulozte koeficienty
% filtru do promenne ‚h':
rlsFilter = dsp.RLSFilter(ORDER, 'ForgettingFactor', LAMBDA);
[yOutput,err] = rlsFilter(y, distortedSound);
h = rlsFilter.Coefficients;
```

```
% Postup pomoci funkce rls:
%
% Postupy vypoctu byly testovane v Matlabu verze 9.2.0.538062 (R2017a).
% Ve starsich verzich nemusi fungovat vypocet pomoci ‚dsp.RLSFilter‘, proto
% je zde uveden i alternativni vypocet impulzni charakteristiky RLS filtru
% pomoci funkce rls (autor: M.H. Hayes):
%
% cd(‚funkce/hayes‘)
% [A,err] = rls(y,distortedSound,ORDER,LAMBDA);
% err = err‘;
% h = A(end,:);
% yOutput = conv(y,h,‘same‘);
% cd(‚../..‘)

% Vykreslete do jednoho grafu ocekavanou casovou radu, casovou radu
% modelovanou pomoci filtru a chybu:
plot(1:length(y), [distortedSound,yOutput,err]);
legend(‚Očekávaný výstup‘, ‚Výstup‘, ‚Chyba‘);
xlabel(‚Čas‘);

% Vykreslete impulzni charakteristiku RLS filtru:
figure, stem(1:ORDER, h); % Obrazek 5D1
title(‚Impulzní charakteristika RLS filtru‘);
xlabel(‚Pořadí vzorku‘);
ylabel(‚Hodnota koeficientu‘);

% Nactete do Matlabu libovolnou melodii a pouzijte na ni adaptovany
% distortion efekt.

% Priklad 1:
handelDistorted = conv(y,h);
sound(handelDistorted)

% Priklad 2:
load gong
sound(y)
gongDistorted = conv(y,h);
sound(gongDistorted)
```



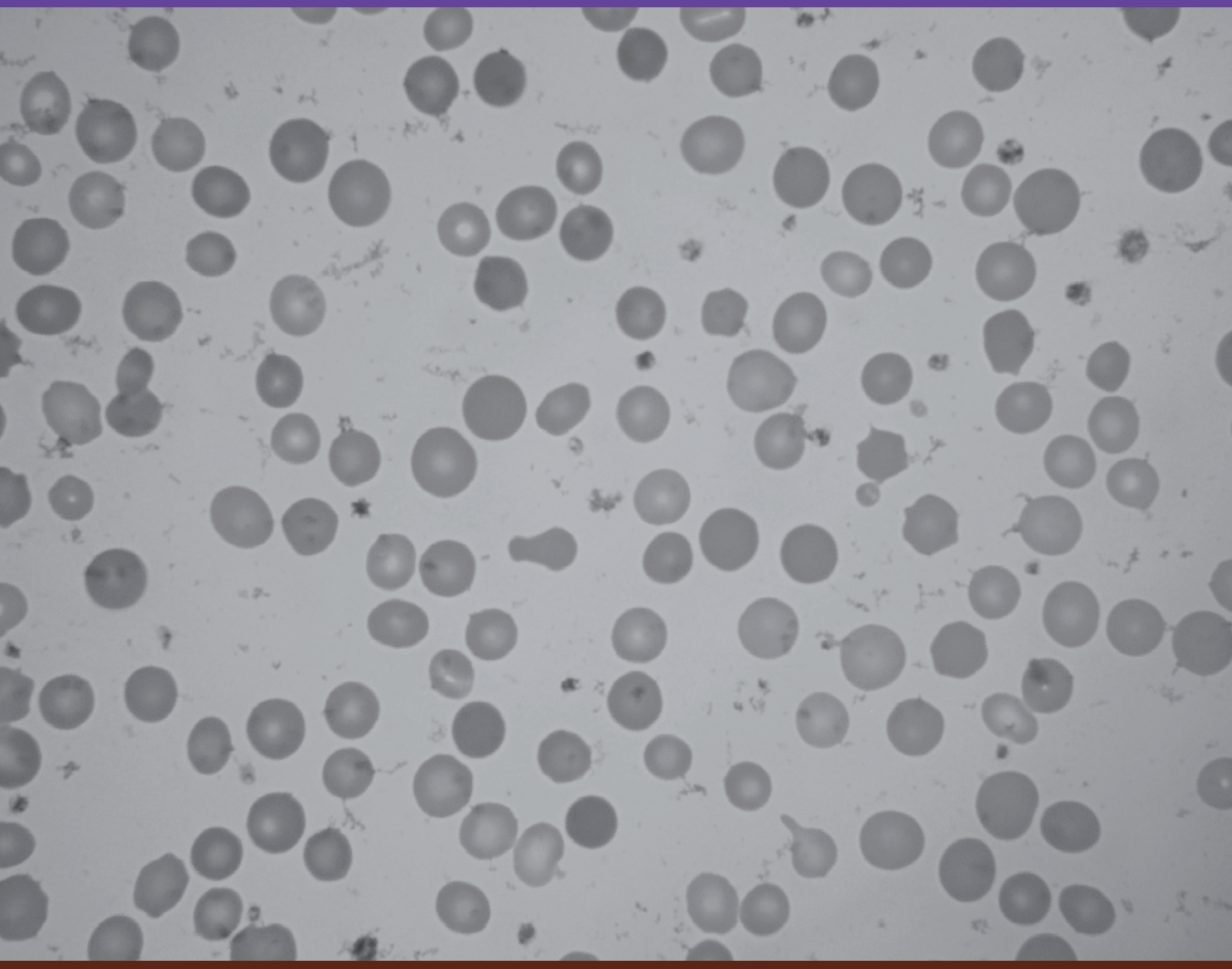
Obrázek 5D1: Impulzní charakteristika RLS filtru

Úkol 2: Odstranění šumu pomocí RLS filtru

Načtěte soubor ‚\data\sum_zvuk.mat' a přidejte ho jako aditivní šum ke zvuku handel. Na rušivou složku aplikujte libovolný filtr nebo vynásobte např. koeficientem 0.2:

```
load(char(strcat(pwd, ‚\data\sum_zvuk.mat')));
load handel
handelNoise = y + 0.2*noise;

% Urcete pocet vzorku impulzni charakteristiky hledaneho filtru:
ORDER = 11; % Napr. 11

% Urcete parametr lambda, ktery urcuje zapominani vlivu vzorku smerem
% do minulosti:
LAMBDA = 1; % Napr. 1

% Navrhnete RLS filtr, na vstup privedte vektor ‚noise' a vektor
% ‚handelNoise' pouzijte jako ocekavany vystupni signal. Vytvorte objekt
% RLS filtru pomoci ‚dsp.RLSFilter' nebo pouzijte funkci ‚rls', kterou
% najdete ve slozce ‚funkce/hayes':
rlsFilter = dsp.RLSFilter(ORDER, ‚ForgettingFactor', LAMBDA);
[~,err] = rlsFilter(noise,handelNoise);
```

```
% Chybova slozka ‚err' pak obsahuje postupne ocisteny zvuk. Poslechnete si
% jej:
sound(err)

% Pozn.: Proste odecteni ruseni nestaci, protoze sum je vynasoben
% koeficientem 0.2 (popripade filtrovan).
sound(handelNoise-noise)

% Postup pomoci funkce rls:
%
% cd(‚funkce/hayes')
% [~,err] = rls(noise,handelNoise,ORDER,LAMBDA);
% sound(err)
```



UNIVERSITAS MASARYKIANA BRUNENSIS

UNIVERSITAS MASARYKIANA BRUNENSIS FACULTAS MEDICA

IBA